SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA,
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Evaluation of Different Bilevel Optimization Algorithms with Applications to Control

DIPLOMA THESIS

FCHPT-5414-28119

Study programme: Automation and Informatization in Chemistry and Food Industry
Study field: 5.2.14 Automation
Workplace: ETH Zurich
Thesis supervisor: Ing. Michal Kvasnica, PhD.
Consultant: Dr. Daniel Axehill

Bratislava 2010 Bc. Ivana Rauová

Slovak University of Technology in Bratislava
Department of Information Engineering and Process Control

Faculty of Chemical and Food Technology
Academic year: 2009/2010
Reg. No.: FCHPT-5414-28119

S T U
F C H P T

DIPLOMA THESIS TOPIC

Student: **Bc. Ivana Rauová**
Student's ID: 28119
Study programme: Automation and Informatization in Chemistry and Food Industry
Study field: 5.2.14 Automation
Thesis supervisor: Ing. Michal Kvasnica, PhD.
Consultant: Dr. Daniel Axehill
Workplace: ETH Zurich

Topic: **Evaluation of Different Bilevel Optimization Algorithms with Applications to Control**

Specification of Assignment:

Bilevel optimization problems are optimization problems where parts of the constraints are in the form that some variables in the problem (the upper level problem) are constrained to be optimal solutions to another optimization problem (the lower level problem). These problems are in most cases non-convex and therefore hard to solve to global optimality. Recently, it has been discovered that these problem formulations can be very useful in automatic control applications. For example, they can be used to compute the worst case deviation of a suboptimal Model Predictive Control scheme compared to the optimal one. The objective with this work is to compare different bilevel optimization algorithms where the comparison is performed both theoretically and numerically. The most important properties compared are if the algorithms can find a global optimizer guaranteed, the numerical robustness (as observed in numerical experiments), and the computational performance.

Thesis length: 40

List of professional works:

1. LOEFBERG, J. YALMIP Wiki. [online]. 2009. URL: http://control.ee.ethz.ch/ joloef/wiki/pmwiki.php.
2. KVASNICA, M. Multi-Parametric Toolbox. [online]. 2009. URL: http://control.ee.ethz.ch/ mpt/.

Assignment procedure from: 15. 02. 2010

Date of thesis submission: 22. 05. 2010

L. S.

Bc. Ivana Rauová
Student

prof. Ing. Miroslav Fikar, DrSc.
Head of office

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

Abstract

The topic of this thesis is bilevel optimization. Bilevel optimization problems are optimization problems where some variables in one optimization problem (the upper level problem) are constrained to be optimal in a second optimization problem (the lower level problem). Recent research has shown that several important problems in for example automatic control can be formulated as bilevel optimization problems. Unfortunately, the bilevel problems are known to often be very hard to solve in practice and the main focus in this thesis is therefore to investigate different algorithms to solve them efficiently. In the thesis, several common types of bilevel problems are discussed. Furthermore, different algorithms to solve these kinds of problems are compared theoretically. Finally, some promising ones are selected to be more thoroughly investigated in numerical experiments on a large number of randomly generated test problems with the purpose to evaluate the computational performance and reliability. In these experiments, bilevel problems with convex quadratic programming problems at the lower level and convex, as well as non-convex, quadratic programming problems at the upper level are considered. Finally, also an automatic control related problem is used as a benchmark problem.

Abstrakt

Predmetom tejto práce je dvoj-úrovňová (bilevel) optimalizácia. Problémy dvoj-úrovňovej optimalizácie sú také problémy, kde niektoré premenné v jednom optimalizačnom probléme (optimalizačný problém hornej úrovne) musia byť optimálne v druhom optimalizačnom probléme (optimalizačný problém dolnej úrovne). Výskum nedávno ukázal, že niekoľko dôležitých optimalizačných problémov, napríklad v oblasti automatického riadenia, môže byť formulovaných ako dvoj-úrovňový problém. Naneštastie, dvoj-úrovňové problémy sa v praxi považujú za výpočtovo veľmi náročné a teda hlavným cieľom tejto práce je preskúmať rozdielne algoritmy, ktoré ich riešia efektívne. V tejto práci je rozobraných niekoľko bežných typov dvoj-úrovňových problémov. Následne, sú v teoretickej rovine rozobraté a porovnané rôzne algoritmy pre riešenie tohto typ problému. Nakoniec, vybrané sľubnejšie vyzerajúce algoritmy sú vyskúšané na veľkom množstve náhodných numerických príkladov s cieľom ukázať efektívnosť z hľadiska výpočtového času a spoľahlivosti. Pre tieto výpočty boli použité dvoj-úrovňové problémy s konvexnými kvadratickými funkciami v probléme dolnej úrovne a konvexnými ako aj nekonvexnými kvadratickými funkciami v probléme hornej úrovne. Nakoniec boli algoritmy odskúšané na štandardnom probléme riadenia.

Contents

List of Symbols and Abbreviations

Symbols, Operators and Functions

Notation	Meaning
$A \succ (\succeq) 0$	A positive (semi)definite matrix
$x \leq (<) y$	Component wise (strict) inequality for vectors x and y
A^T	Transpose of matrix A
$I, (I_n)$	Identity matrix (with n rows)
$\mathbf{1}, (\mathbf{1}_n)$	A vector (with n components) with all component equal to 1
$\underset{x}{\operatorname{argmin}} f(x)$	The optimal solution to $\min_x f(x)$
$\operatorname{dom} f(x)$	Domain of function f
$\mathcal{L}$	Lagrangian
$\lfloor x \rfloor$	The floor function. Gives the largest integer less then or equal to x
$\lceil x \rceil$	The ceiling function. Gives the smallest integer grater then or equal to x
∇f	Gradient of a function f
$\otimes$	Conjunction

Sets

Notation	Meaning
$\mathbb{R}$	Set of real numbers

$\mathbb{R}^{n\times 1}$	Set of real vectors with n components
$\mathbb{R}^{n\times m}$	Set of real matrices with n rows and m columns
$\mathbb{S}^n$	Set of symmetric matrices with n rows
$\{0,1\}^n$	Set of vectors with n binary variables

Abbreviations and Acronyms

Abbreviation	**Meaning**
B&B	Branch-and-Bound method
BLPP	Bilevel Programming Problem
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MIQP	Mixed Integer Quadratic Programming
MPC	Model Predictive Control
QP	Quadratic Programming
SQP	Sequential Quadratic Programming

Introduction

Bilevel optimization problems are optimization problems where some variables in one optimization problem (the upper level problem) are constrained to be optimal in a second optimization problem (the lower level problem). They involve a hierarchy of two optimization problems. Their application design optimization in process systems engineering or management and in automatic control, what is important for us. Unfortunately, bilevel problems are hard to solve by common algorithms which requires development of new ones which are able to solve these problem. An important property of these algorithms is the ability to reach a global optimum which is often very hard in these problems. Many algorithms are able to reach only a local optimum. It is enough for some problems but some of them require a globally optimal solution. In the automatic control one can find convex problems, which means that both objective functions, in the lower and upper problem are convex quadratic functions. Moreover, many important problems exist in non-convex form where the upper level optimization problem consists of a nonconvex quadratic objective function with linear constraints while the lower level problem has a convex quadratic objective function.

Many algorithms to solve bilevel problems have been developed and published in the literature. They were compared with some well-known algorithms, e.g. the algorithm of Bard and Moore [20]. It is necessary to compare the type of the problem which they are able to solve, way they deal with the problem, how they transform it with aim to make problem easier, efficiency of the algorithm in term of the computation time and if they are able to compute solutions that are guaranteed to be globally optimal. It is also interesting to test the algorithms in numerical experiments which has been done as a part of this thesis.

Chapter 1

Optimization Preliminaries

This chapter will be opened with basic definitions for optimization. Furthermore, descriptions of typical optimization problems will be given.

1.1 Optimization

Optimization, from the mathematical point of view, is a procedure which focuses on choosing the best element from a set of possibly good alternatives. The objective is to minimize or maximize a function by repeatedly choosing the values of proper type within an allowed set. In other words, the objective is to find an optimal solution for the objective function within the set given by restrictions called constraints. Depending on the formulation of the problem one can divide optimization problems into different types, e.g. convex optimization, non-convex optimization, linear programming, etc. see Fig. 1.1.

1.2 Basic Concepts

In this section will important concepts in the convex and non-convex optimization as convex functions and sets will be presented.

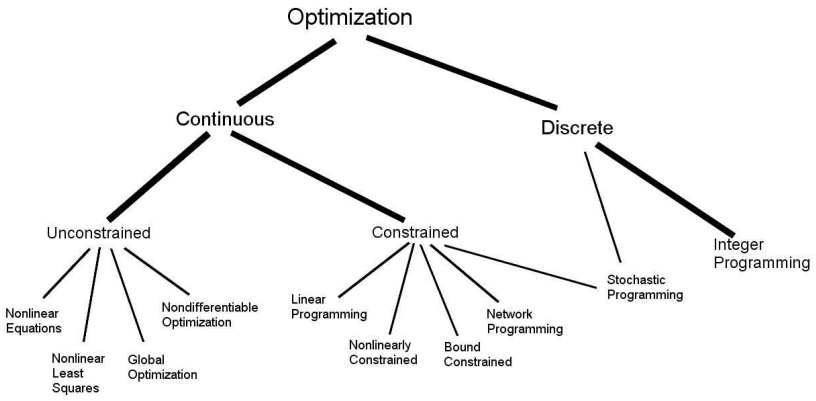


Figure 1.1: Division of the optimization problems

1.2.1 Definitions

Definition 1.1 (Convex set) *[39] A set $\mathbb{R}_c \subset \mathbb{E}^n$ is said to be convex if for every pair of points x_1,$x_2 \subset R_c$ and for every real number α in the range $0 \leq \alpha \leq 1$, the point*

$$x = \alpha x_1 + (1 - \alpha)x_2$$

is located in $\mathbb{R}_c$, i.e., $x \in \mathbb{R}_c$.

In other words, set $\mathbb{R}_c$ is convex when any two points x_1,$x_2 \in \mathbb{R}_c$ are connected by a straight line and each segment of this line lies within the set $\mathbb{R}_c$. If some points of the line are not in the $\mathbb{R}_c$, this set is called *non-convex*. Convexity of the sets is depicted in the Fig. 1.2.

Definition 1.2 (Convex function) *[39] A function $f(x)$ defined over a convex set R_c is said to be convex if for every pair of points x_1,$x_2 \in R_c$ and every real number α in the range $0 < \alpha < 1$, the inequality*

$$f[\alpha x_1 + (1 - \alpha)x_2] \leq \alpha f(x_1) + (1 - \alpha)f(x_2)$$

holds. If $x_1 \neq x_2$ and

$$f[\alpha x_1 + (1 - \alpha)x_2] < \alpha f(x_1) + (1 - \alpha)f(x_2)$$

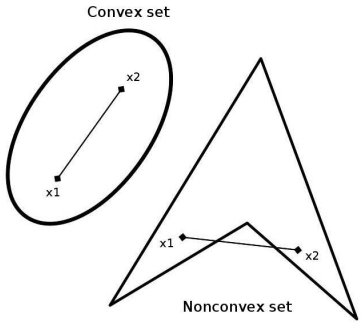


Figure 1.2: Convexity of the sets

then $f(x)$ is said to be strictly convex.

Definition 1.3 (Concave function) *[39] If $\phi(x)$ is defined over a convex set R_c and $f(x) = -\phi(x)$ is convex, then $\phi(x)$ is said to be concave. If $f(x)$ is strictly convex, $\phi(x)$ is strictly concave.*

Definition 1.4 (Smooth function) *If the second derivative of the function $f(x)$ exists and is continuous at every point x, the function $f(x)$ is smooth.*

1.2.2 General Problem Description

In this thesis, an optimization problem in the form

$$\min_{x} \quad f(x) \tag{1.1a}$$

$$\text{s.t.} \quad g_i(x) \leq 0 \qquad i = 1, \ldots, m \tag{1.1b}$$

$$h_j(x) = 0 \qquad j = 1, \ldots, p \tag{1.1c}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$, $g_i(x) : \mathbb{R}^n \to \mathbb{R}$ and $h_j(x) : \mathbb{R}^n \to \mathbb{R}$ is said to be in a *standard form*. The function $f(x)$ is called the cost function, or objective function, $g_i(x)$, $i = 1, \ldots, m$ represent the inequality constraint functions and $h_i(x)$, $i = 1, \ldots, p$ denote the equality constraint functions. If $m = p = 0$ the problem is said to be *unconstrained*. The domain of the optimization problem is given by an intersection of the domains of the objective

function and constraint functions [5].

$$\mathcal{D} = \bigcap_{i=1}^{m} \text{dom } g_i \cap \bigcap_{j=1}^{p} \text{dom } h_j \tag{1.2}$$

If a point $x \in \mathcal{D}$ satisfies all the equalities and inequalities in (1.1), it is said to be a *feasible point*. If there exists one such point then the problem is said to be *feasible*, otherwise the problem is said to be *infeasible*. The feasible point x^* is an optimal solution of the problem (1.1). It can be a *local minimizer* if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{N}$. A point x^* is *global minimizer* if $f(x^*) \leq f(x)$ for all feasible x [37].
A special case of (1.1) is when the objective function (1.1a) and inequality constraint functions (1.1b) are convex and the equality constraint functions (1.1c) are affine, that is, $h_i(x) = a_i^T x - b_i$. In that case the problem in (1.1) is called a *convex optimization problem*. For this kind of problems the fundamental property is that a local minimizer is also a global minimizer. In other words a locally optimal solution is also a globally optimal solution.
The set in which the problem (1.1) have a feasible solutions is called the *feasible set*

$$P = \{x \in \mathbb{R}^n : g_i(x) \leq 0, h_j(x) = 0\} \tag{1.3}$$

In a case of empty set we are speaking about an *infeasible set*. This means, that no solution for the problem (1.1) can be obtained. The problem is said to be *unbounded* when the objective function is unbounded below on the feasible set. This gives evidence that solution for the problem can not be found. Size of the problem must be strictly proper ($p < n$) otherwise system (1.1c) contains redundant rows or is infeasible [37]. In the other case ($p \geq n$) some kind of factorization, e.g. QR or LU [34], has to be used to obtain a system (1.1c) with full row rank matrices.

1.3 Optimality Conditions

The Karush-Kuhn-Tucker (KKT) optimality conditions are fundamental for many algorithms for constrained optimization problems. They can also be called *first-order conditions* because in their derivation properties of the gradients are used.

The Lagrangian function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with (1.1) plays an important role in the derivation of the KKT conditions. The Lagrangian is defined as

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) \tag{1.4}$$

where λ_i and μ_j are Lagrangian multipliers associated with the i^{th} inequality constraint and j^{th} equality constraint, respectively. The vectors of Lagrangian multipliers $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^p$ are called the *dual variables* for the problem in (1.1). For the optimal triple (x^*,λ^*,μ^*) Theorem 1.1 must be satisfied.

Theorem 1.1 *[5] Assume that the problem (1.1) is convex, $g_i(x)$, $i = 1, \ldots, m$ and $h_j(x)$, $j = 1, \ldots, p$ are differentiable and strong duality holds (see Theorem 1.2). Then the so-called KKT conditions are necessary and sufficient conditions for x^* and (μ^*, λ^*) to be primal and dual optimal points*

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} \mu_i^* \nabla h_i(x^*) = 0 \tag{1.5a}$$

$$g_i(x^*) \leq 0 \qquad i = 1, \ldots, m \tag{1.5b}$$

$$h_j(x^*) = 0 \qquad j = 1, \ldots, p \tag{1.5c}$$

$$\lambda_i^* g_i(x^*) = 0 \qquad i = 1, \ldots, m \tag{1.5d}$$

$$\lambda_i^* \geq 0 \qquad i = 1, \ldots, m \tag{1.5e}$$

Proof: See [37, p.329]

In Theorem 1.1, the equation (1.5a) represents *stationarity condition*, (1.5e) represents *dual feasibility*, the last two equations (1.5b)- (1.5c) represent *primal feasibility* and the equation (1.5d) is *complementarity slackness*. In (1.5d) either $\lambda_i = 0$ or $g_i(x) = 0$ satisfies the equality. If $g_i(x) < 0$ we are speaking about *inactive constraint i*, in the other case, $g_i(x) = 0$, we are speaking about *active constraint i*.

If the objective function is a convex function and the feasible set is convex, the KKT conditions are sufficient to reach a global minimizer. In the case of a non-convex problem sufficiency of the KKT conditions is no longer guaranteed.

1.4 Linear Programming

Linear programming (LP) is a special case of the convex optimization. The problem is defined as to minimize a linear objective function subject to linear equality and inequality constraints. The standard form of the problem

$$\min_{x} \quad c^T x \tag{1.6a}$$

$$\text{s.t.} \quad A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{1.6b}$$

$$x \geq 0 \tag{1.6c}$$

where c,$x \in \mathbb{R}^{n\times 1}$, $b_{\mathcal{E}} \in \mathbb{R}^{m\times 1}$ and $A_{\mathcal{E}} \in \mathbb{R}^{m\times n}$. Graphical representation of the LP problem is depicted in the Fig. 1.3.

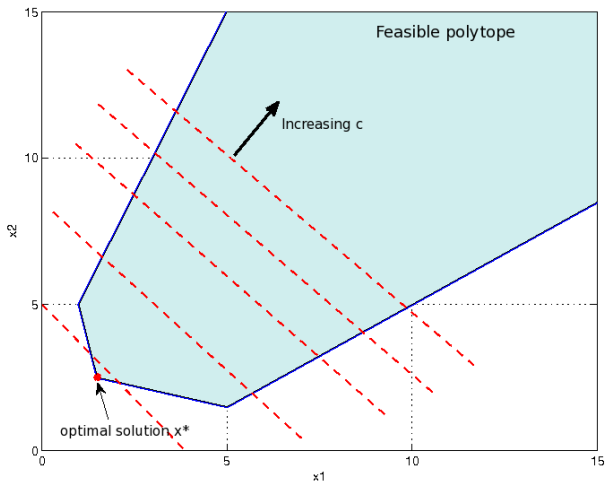


Figure 1.3: Optimal solution x^* in a two dimensional linear programming problem

While there are different kinds of linear problems, each of them can be written in the form in (1.6). Even if the problem contains inequalities, *slack variables* z can be used to transform the problem into the form:

$$\min_{x} \quad c^T x \tag{1.7a}$$

$$\text{s.t.} \quad A_{\mathcal{E}} x + z = b_{\mathcal{E}} \tag{1.7b}$$

$$z \geq 0 \tag{1.7c}$$

Assuming the matrix A_ϵ from (1.6b) has full row rank, it can be divided as $A_\epsilon = [B, N]$ where B is a matrix with linearly independent columns. Then the linear system $Bx_b = b_\epsilon$ has a unique solution. If $x = (x_b, 0)$ then $Ax = b$ and $x = (x_b, 0)$ is called a *basic solution*. Then components of x which belongs to the matrix B are called *basic variables* [34]. However, if one or more basic variables in the basic solution are zeros, the solution of the problem (1.6) is said to be a *degenerate solution*. Finally, the *basic feasible solution* is a basic solution which takes place in the feasible set, i.e. solution is feasible ($x \geq 0$).

1.4.1 Optimality and Duality

Relaxation of the problem in (1.6) can be derived using Lagrangian relaxation. Considering convexity of the problem, optimality conditions are sufficient for reaching a global minimum [37]. Using the definition in (1.4) the Lagrangian function of (1.6) is in the form

$$\mathcal{L}(x, \mu, \lambda) = c^T x - \mu^T (A_\mathcal{E} x - b_\mathcal{E}) - \lambda^T x \tag{1.8}$$

and then necessary conditions for x^* to be a solution of (1.6) are that there exist vectors λ and μ such that

$$A_\mathcal{E}^T \mu + \lambda = c \tag{1.9a}$$

$$A_\mathcal{E} x = b_\mathcal{E} \tag{1.9b}$$

$$x \geq 0 \tag{1.9c}$$

$$\lambda \geq 0 \tag{1.9d}$$

$$x_i \lambda_i = 0 \quad i = 1,2,\ldots,m \tag{1.9e}$$

Consider a vector triple (x^*,μ^*,λ^*) that satisfies (1.9). Then it holds that

$$c^T x^* = (A_\mathcal{E}^T \mu^* + \lambda^*)^T x^* = (A_\mathcal{E} x^*)^T \mu^* = b_\mathcal{E}^T \mu^* \tag{1.10}$$

which brings us to the *dual problem* where the objective functions $c^T x^*$ and $b_\mathcal{E}^T \mu^*$, for the primal and dual problem respectively, have the same optimal value. The dual problem is

closely related to the primal one and can be written as

$$\max_{\mu} \quad b_{\mathcal{E}}^T \mu \tag{1.11a}$$

$$\text{s.t.} \quad A_{\mathcal{E}}^T \mu \leq c \tag{1.11b}$$

and by introducing *slack variables* s problem (1.11) can be reformulated as

$$\max_{\mu} \quad b_{\mathcal{E}}^T \mu \tag{1.12a}$$

$$\text{s.t.} \quad A_{\mathcal{E}}^T \mu + s = c \tag{1.12b}$$

$$s \geq 0 \tag{1.12c}$$

The Lagrangian function to the problem in the problem (1.11) is

$$\overline{\mathcal{L}}(\mu, x) = -b_{\mathcal{E}}^T \mu^T - x^T (c - A_{\mathcal{E}} \mu) \tag{1.13}$$

where $x \in \mathbb{R}^n$ denotes the Lagrange multipliers for the inequality constraints in the dual problem. The necessary conditions (or KKT conditions) are in the form

$$A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{1.14a}$$

$$A_{\mathcal{E}}^T \mu \leq c \tag{1.14b}$$

$$x \geq 0 \tag{1.14c}$$

$$x_i (c - A_{\mathcal{E}}^T \mu)_i = 0 \quad i = 1,2,\ldots,n \tag{1.14d}$$

With the primal-dual problem, there are connected terms “weak ”and “strong ”duality. Weak duality means that for $c^T x \geq b_{\mathcal{E}}^T \mu$ (that is, the dual objective is a *lower bound* on the primal objective) when both the primal and dual variables are feasible [37]. Strong duality is fundamental to the theory of LP and is described by the Theorem 1.2

Theorem 1.2

(i) If either problem (1.6) or (1.11) has a (finite) solution, then so does the other, and the objective values are equal.

(ii) If either problem (1.6) or (1.11) is unbounded, then the other problem is infeasible.

Proof: See [37, p.361]

In the complementarity conditions (1.9e) and (1.14d) it is easy to see that at least one of the components x_i and s_i or x_i and $(c - A_{\mathcal{E}}^T \mu)_i$ must be equal to zero to satisfy the condition.

The most commonly used methods are the Simplex Method and Interior Point Methods. These methods are deeply described and discussed in [37, 34].

1.5 Quadratic Programming

Quadratic programming (QP) is a class of problems belonging to the class of constrained nonlinear optimization problems. The problem is described by the quadratic objective function subject to linear constraints in a form of equalities and inequalities. For the quadratic programming a *standard form* is stated as

$$\min_x \quad \frac{1}{2} x^T H x + c^T x \tag{1.15a}$$

$$\text{s.t.} \quad A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{1.15b}$$

$$A_{\mathcal{I}} x \leq b_{\mathcal{I}} \tag{1.15c}$$

where c,$x \in \mathbb{R}^{n \times 1}$, H is symmetric Hessian matrix of dimension $n \times n$, rows in $A_{\mathcal{E}} \in \mathbb{R}^{p \times n}$ are given by $\{a_i \in \mathbb{R}^{1 \times n} | i \in \mathcal{E}\}$ and rows in $A_{\mathcal{I}} \in \mathbb{R}^{m \times n}$ are given by $\{a_i \in \mathbb{R}^{1 \times n} | i \in \mathcal{I}\}$. The vectors $b_{\mathcal{E}}$ and $b_{\mathcal{I}}$ are defined in the same way. The sets $\mathcal{E}$ and $\mathcal{I}$ represent finite sets of indices. A graphical representation of the problem (1.15) is depicted in the Fig. 1.4.

Depending on the definiteness of the matrix H, QP problems can be divided into two classes. The first class is *convex QP* (*strictly convex QP*) when $H \succeq 0$ ($H \succ 0$) and the second one is *non-convex QP* when $H \not\succeq 0$.

The Lagrangian for the problem in (1.15) is defined as

$$\mathcal{L}(x, \mu, \lambda) = \frac{1}{2} x^T H x + c^T x + \lambda^T (A_I x - b_I) + \mu^T (A_{\mathcal{E}} x - b_{\mathcal{E}}) \tag{1.16}$$

Then necessary conditions for x^* to be a solution of (1.15) are that there exist vectors

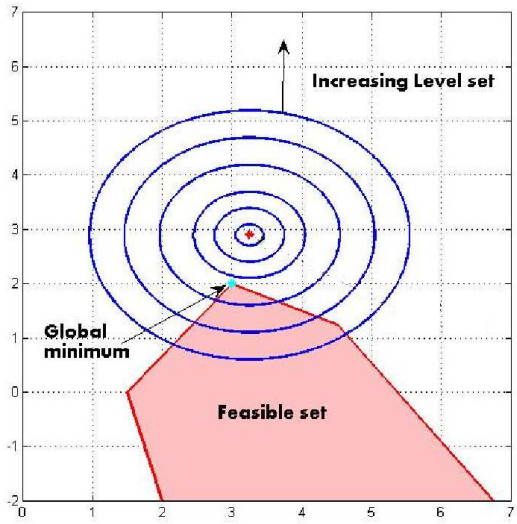


Figure 1.4: Optimal solution x^* in a two dimensional quadratic programming problem

$\mu \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p$ such that the following conditions are satisfied

$$Hx + A_{\mathcal{E}}^T \mu + A_I^T \lambda = -c \tag{1.17a}$$

$$A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{1.17b}$$

$$A_I x \leq b_I \tag{1.17c}$$

$$\lambda \geq 0 \tag{1.17d}$$

$$\lambda_i (A_I x - b_I)_i = 0 \ , i = 1, \ldots, m \tag{1.17e}$$

Also in this case active set methods can be employed. For solving QPs there exist methods like Interior points method or Active-set methods which are described in [37].

1.6 Mixed Integer Linear Programming

In mixed-integer programming (MIP) problems some of the variables are real-valued and some of them are integer valued. Therefore the model is "mixed". If the objective function and constraints are in a linear form, the problem is called a mixed-integer linear program-

ming (MILP) problem. After introducing binary variables into the LP problem (1.6), the *standard form* of an MILP is given as

$$\min_x \quad c^T x \tag{1.18a}$$

$$\text{s.t.} \quad A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{1.18b}$$

$$x \geq 0, x_i \in \{0,1\}, \forall i \in \mathbb{B} \tag{1.18c}$$

where $n = n_c + n_b$ and $c \in \mathbb{R}^{n\times 1}$, $A_{\mathcal{E}} \in \mathbb{R}^{m\times n}$, $b_{\mathcal{E}} \in \mathbb{R}^{m\times 1}$ and $x \in \mathbb{R}^{n\times 1}$ and $\mathbb{B}$ denotes the set of indices for the binary variables.

MILP problems are usually solved using branch-and-bound algorithms [22]. There exist several software for solving MILP problems, e.g. free software like Yalmip or milp.m in MATLAB and commercial software like CPLEX.

1.7 Mixed Integer Quadratic Programming

The mixed-integer quadratic programming (MIQP) problem is a special case of mixed-integer nonlinear programming (MINLP). Like in the MILP problem, the MIQP problem is allowed to contain integer variables as well as the real-valued ones. Then the MIQP can be derived from (1.15) involving binary variables, which leads to the *standard form*

$$\min_x \quad \frac{1}{2} x^T H x + c^T x \tag{1.19a}$$

$$\text{s.t.} \quad A_{\mathcal{E}} x = b_{\mathcal{E}} \tag{1.19b}$$

$$A_{\mathcal{I}} x \leq b_{\mathcal{I}} \tag{1.19c}$$

$$x_i \in \{0,1\}, \forall i \in \mathbb{B} \tag{1.19d}$$

where $n = n_c + n_b$, $c \in \mathbb{R}^{n\times 1}$ and $x \in \mathbb{R}^{n\times 1}$. There $\mathbb{B}$ denotes the set of indices for binary variables. Let $A_{\mathcal{E}}$,$A_{\mathcal{I}}$,$b_{\mathcal{E}}$ and $b_{\mathcal{I}}$ to be defined as in (1.15). The most commonly used methods for this kind of problems are, [23]:

- Cutting plane methods
- Decomposition methods

- Logic-based methods
- Branch and bound methods

where branch-and-bound methods are the most commonly used methods. Example of software that can be used to solve MIQP problems in MATLAB is, mpqp.m or the free software Yalmip. A well-known commercial software is CPLEX.

Chapter 2

Bilevel Programming

This chapter is opened with a description of bilevel programming problem and its basics. Furthermore, a theoretical comparison of different algorithms from the literature will be done.

2.1 Problem Description

An optimization problem constrained by another optimization problem is called a bilevel programming problem (BLPP). From the mathematical point of view it is a problem with hierarchical structure where two independent decision-makers appear. One can consider this problem as a sequential game, which has its origin in the *Stackelberg game theory* [38]. That is, optimal reaction vector y of the second player ("follower") is included in the decision making of the first player ("leader"). Leader's reaction vector is denoted as x. This

rule can be represented in a mathematical form of the bilevel program

$$\min_{x} \; F(x, y(x)) \tag{2.1a}$$

$$\text{s.t.} \; G(x, y(x)) \leq 0 \tag{2.1b}$$

$$H(x, y(x)) = 0 \tag{2.1c}$$

$$y(x) = \underset{y}{\operatorname{argmin}} \; f(x, y) \tag{2.1d}$$

$$\text{s.t.} \; g(x, y) \leq 0 \tag{2.1e}$$

$$h(x, y) = 0 \tag{2.1f}$$

Equations (2.1a) - (2.1c) describe the leader's decision subproblem, while (2.1d) - (2.1f) represent the follower's decision subproblem. For the comprehensive description, $F(x, y)$ and $f(x, y)$ are the leader's and follower's objective functions, respectively. Further, $G(x, y)$ and $g(x, y)$ represent inequality constraint functions in the leader's and follower's subproblem, respectively, and $H(x, y)$ and $h(x, y)$ are equality constraint functions in leader's and follower's subproblem, respectively. To build up a complete picture of the problem, it is necessary to define some more terms which will be used for what follows. The *relaxed feasible set* for BLPP is defined as $\Omega = \{(\mathbf{x}, \mathbf{y}) : \mathbf{G}(\mathbf{x}, \mathbf{y}) \leq 0, \mathbf{H}(\mathbf{x}, \mathbf{y}) = 0, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0, \mathbf{h}(\mathbf{x}, \mathbf{y}) = 0\}$. The *follower's feasible set* for the decision variables $\mathbf{x}$ of the leader's problem defined for every $x \in \mathcal{X}$ is specified as $\Omega(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} \in \mathcal{Y}, \mathbf{h}(\mathbf{x}, \mathbf{y}) = 0, \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0\}$ while the *rational follower's reaction set* for every $x \in \mathcal{X}$ is defined as $\mathcal{M}(x) = \{\mathbf{y} : \mathbf{y} \in \operatorname{argmin}\{f(\mathbf{x}, \mathbf{y}) : \mathbf{y} \in \Omega(\mathbf{x})\}\}$. Finally, it is essential to define the *inducible region* or *feasible set for BLPP* as $\mathcal{IR} = \{(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) \in \Omega, \mathbf{y} \in \mathcal{M}(\mathbf{x})\}$. This is the set over which the leader may optimize. All the definitions assume a bounded space, where the optimal solution can be found.

2.1.1 Relaxation of Bilevel Programming Problems

Using necessary optimality conditions (1.5) and assuming validity of a certain regularity assumption the BLPP can be replaced with

$$\min_{x,y,\lambda,\mu} \quad F(x,y) \tag{2.2a}$$

$$\text{s.t.} \quad G(x,y) \leq 0 \tag{2.2b}$$

$$H(x,y) = 0 \tag{2.2c}$$

$$\nabla_y f(x,y) + \lambda^T \nabla_y g(x,y) + \mu^T \nabla_y h(x,y) = 0 \tag{2.2d}$$

$$\lambda_i g_i(x,y) = 0, \quad i = 1,\ldots,m \tag{2.2e}$$

$$\lambda \geq 0 \tag{2.2f}$$

$$g(x,y) \leq 0 \tag{2.2g}$$

$$h(x,y) = 0 \tag{2.2h}$$

Problem (2.1) is said to be relaxed in a form (2.2) when the optimality conditions are not necessary and sufficient but strong duality still holds. The problem (2.1) is said to be equivalent with a form (2.2) for linear or convex quadratic follower's problems.
By inserting (1.6) into (2.1) we obtain convex BLPP with the linear objective function

$$\min_{x} \quad c^T x + d^T y \tag{2.3a}$$

$$\text{s.t.} \quad A_{\mathcal{I}} x - E_{\mathcal{I}} y - b_{\mathcal{I}} \leq 0 \tag{2.3b}$$

$$x \geq 0 \tag{2.3c}$$

$$\min_{z} \quad e^T y + f^T x \tag{2.3d}$$

$$\text{s.t.} \quad F_{\mathcal{I}} y - G_{\mathcal{I}} x - h_{\mathcal{I}} \leq 0 \tag{2.3e}$$

$$y \geq 0 \tag{2.3f}$$

while inserting (1.15) into (2.1) results in BLPP with quadratic objective function where H_1 can be $H_1 \succeq 0$ or $H_1 \not\succeq 0$ and $H_2 \succeq 0$ for non-convex and convex problem, respectively.

$$\min_{x} \ \frac{1}{2}x^T H_1 x + c^T x + d^T y \tag{2.4a}$$
$$\text{s.t.} \ \ A_{\mathcal{I}} x - E_{\mathcal{I}} y - b_{\mathcal{I}} \leq 0 \tag{2.4b}$$
$$A_{\mathcal{E}} x - E_{\mathcal{E}} y - b_{\mathcal{E}} = 0 \tag{2.4c}$$
$$\min_{z} \ \frac{1}{2}y^T H_2 y + e^T y + f^T x \tag{2.4d}$$
$$\text{s.t.} \ \ F_{\mathcal{I}} y - G_{\mathcal{I}} x - h_{\mathcal{I}} \leq 0 \tag{2.4e}$$
$$F_{\mathcal{E}} y - G_{\mathcal{E}} x - h_{\mathcal{E}} = 0 \tag{2.4f}$$

Bilevel programming belongs to the class of $\mathcal{NP}$-hard problems, which loosely means that no *polynomial time algorithm* exist for solving it unless $\mathcal{P} = \mathcal{NP}$ [17]. It was proven that linear bilevel programming [35] and quadratic bilevel programming [14] are $\mathcal{NP}$-hard as well as finding their local optimum. This is the reason why there do not exist many effective algorithms for solving these problems. Effectivity of reaching the optimal solution depends on the form of the problem (2.1). If (2.3) or (2.4) with $H_1, H_2 \succ 0$ is employed, bilevel problem is said to be convex problem with linear constraints. In case of (2.4) with $H_1 \not\succeq 0$ and $H_2 \succ 0$, bilevel problem is non-convex with linear constraints. In this thesis both convex problems with linear constraint functions and non-convex problem with linear constraints in the leader's subproblem will be considered. However, the follower's problem will always be assumed convex.

For solving linear and convex BLPP you can find different algorithms. One possible partition of these algorithms is, [36]:

- Branch-and-bound algorithms
- Cutting plane algorithms
- Barrier and penalty functions algorithms
- Descent algorithms
- Trust region algorithms

- SQP algorithms
- Heuristics algorithms
- Algorithms for solving follower's subproblem explicitly (e.g. mp-LP, mp-QP)
- DC programming
- Genetic algorithms

Algorithms based on these methods will be compared in the sequel.

2.1.2 Big-M Formulation

One way of modeling that a constraint can be active or inactive is a so-called *big-M formulation*. In this approach, a large positive value (scalar variable M) is added in each constraint (2.2f) and (2.2f) and is multiplied with a binary variable δ. Equations for one line of (2.2e) are as follows

$$0 \leq \lambda_i \leq M\delta_i \tag{2.5a}$$

$$-M(1-\delta_i) \leq g_i(x) \leq 0 \tag{2.5b}$$

where i denotes actually computed constraint and $\delta_i \in \{0,1\}$. Now one can describe this as an IF-THEN-ELSE relation

IF $\delta_i = 1$ THEN
$0 \leq \lambda_i$ and $g_i(x) = 0 \Rightarrow$ constraint i is active
ELSE
$\lambda_i = 0$ and $g_i(x) \leq 0 \Rightarrow$ constraint i is inactive

This method belongs to those that are used very often, and its drawback is in including many new binary variables, one for each constraint. Another disadvantage is the selection of the value for M. Especially dual variables are extra hard since bounds are not known like in the case of inequality constraints (2.1e) which often can be computed from the problem

formulation. If one chooses a very small value (e.g. $M = 10$), the optimal solution in the original problem has been cut away and is not feasible anymore after the big M formulation. On the other side, with very large value (e.g. $M = 10^9$) one can get numerical problems and not obtain the solution or the time necessary to obtain the solution could be very long. Proper value of M is determined by experiments.

2.2 Survey of Bilevel Programming Algorithms

In this section some of the algorithms will be briefly described and compared in a theoretical way. The main objective is to find similarities and differences between them.

2.2.1 Branch-and-Bound Algorithms

In this group of algorithms, the branch-and-bound (B&B) method is employed to systematically enumerate all candidate solutions, where large subsets of fruitless candidates are discarded by using upper and lower bounds on the objective function value, [40]. A closer description of the B&B algorithm can be found in [4, 9, 22]

Algorithm 1: [10]

The algorithm described in [10] is applicable to the linear BLPP (2.3). The main principle of the method is an introduction of binary variables α connected to each inequality constraint in the follower problem's constraints (e.g. page 1204 in [10]). These boolean variables determine whether the constraint holds with equality or not.
Concrete steps can be found on the page 1202 in [10]. The numerical results and computation times were compared with those obtained by the algorithms of Bard and Moore [20] and Judice and Faustino [1]. Bard's algorithm was outperformed by Algorithm 1. In the latter the same result was obtained in roughly the same computation time but on a less powerful computer.

Algorithms 2 and 3: [18]

The algorithms described in [18] are used to solve nonlinear BLPP and can guarantee a global solution only if the follower's objective function (2.1d) is strictly convex in y for all $y \in \Omega(x)$ and also all functions (2.1d) - (2.1f) are twice continuously differentiable. Furthermore, the leader's objective function (2.1a) must be continuous and convex in x and y. To make problem solvable, the KKT conditions (1.5) are used for reformulation and in this case conditions are changed. The final form of the problem is a one-level problem with constraints in non-linear and non-convex form like in (2.2).
In Algorithm 2 a combination of depth-first-search and breath-first-search is used to enumerate solution possibilities (nodes in a tree) which are then reduced by using implicit enumeration (e.g. B&B method). Concrete steps can be found on the page 85 in [18]. This algorithm performed well in terms of CPU time (less than 400 CPU s).
Algorithm 3 finds an ϵ-optimal solution to the general nonlinear BLPP using bisection method [34]. The steps can be found on the page 86 in [18]. It is necessary for the follower's subproblem to be convex. Numerical studies showed that Algorithm 3 performed better without lower bounds on the follower's variables because numerical difficulties appeared when bounds were included. Next drawback is that the algorithm is time consuming in sense that it attempts to solve subproblems even if they are infeasible.

Algorithm 4: [3]

In the algorithm in [3], f and g are assumed to be smooth and convex, h is linear in y at fixed x and smooth. The following problem types can be solved by the algorithm:

- linear follower's problem
- convex follower's problem
- fractional bilevel problem with constraints $\Rightarrow$ non-convex follower's problem
- nonlinearities and non-convexities in both subproblems

Using the KKT conditions (1.5), the problem is changed into an equivalent one described by (2.2). Note that the problem is nonlinear and non-convex due to the stationarity conditions (2.2d) and the complementarity constraints (2.2e).

To solve the problem, first an underestimation is necessary to make, afterwards implicit enumeration methods,e.g. B&B, can be employed. The one-level problem (2.2) is assumed to satisfy a linear independence condition. An active set strategy is used for discrete decisions on the set of active constraints. For global optimization of nonlinear mixed-integer problems the Special Structure Mixed Integer Nonlinear αBB, SMIN-αBB algorithm (a deterministic global optimization method based on a B&B framework) or the Global Structure Mixed Integer Nonlinear αBB, GMIN-αBB algorithm can be used. Detailed description of both algorithms is in [2].

Examples of such problems can be found on the pages 15-23 in [3] and concrete steps of the algorithm are described on the page 11.

Algorithm 5: [16]

The algorithm described in [16] is applicable to a convex BLPP. Functions F is assumed to be convex and H, G are assumed to be non-convex in all their arguments, g is non-convex and f is strictly convex in y for fixed x and all the functions are smooth. The problem should be solved after reformulation of the KKT conditions in (1.5) and employing an active set strategy. Steps can be found on the page 20 in [16]. A global optimal solution is guaranteed only when F is strictly convex in (x, y), f is quadratic in (x, y) and g is affine. The numerical results were compared with the barrier method of Aiyoshi and Shimizu [21] and GRG2 of Lasdon et al. [25]. Both algorithms were outperformed by Algorithm 5 in the amount of solved problems. An example of the problem and its solution can be found on page 23 in [16].

2.2.2 Cutting Plane Algorithms

Cutting-plane methods are optimization methods which iteratively refine a feasible set or objective function by means of linear inequalities, called cuts. Such procedures are commonly used to find integer solutions to MILP problems (section 1.6), as well as to solve general, not necessarily differentiable convex optimization problems [41].

Algorithm 6: [31]

You can find a thorough description of this cutting plane algorithm in [31]. The main principle is that the original problem is relaxed by ignoring the follower's minimization. Introducing new slack variables, the feasible set is defined by a new marginal function which stands to be concave on $\mathcal{M}(x)$ and is constructed by the complementarity conditions (2.2e) for given values x and y. One of the steps of the algorithm on the page 130 in [31] is the use of the Tuy's cut and the Valid cut. During the procedure each cut should eliminate as much as possible of the unnecessary part of the feasible region and new vertices are generated. This continues until an optimal solution is found.
Numerically, the algorithm was tested only on small examples found in articles. No comparison with other algorithms was made.

2.2.3 Barrier and Penalty Function Algorithms

Barrier and penalty methods are a certain classes of algorithms for solving constrained optimization problems. A penalty method replaces a constrained optimization problem by a series of unconstrained problems whose solutions ideally converge to the solution of the original constrained problem. The unconstrained problems are formed by adding a term to the objective function that consists of a penalty parameter and a measure of violation of the constraints. The measure of violation is nonzero when the constraints are violated and is zero in the region where constraints are not violated, [42]. Basics of these methods can be found in [37].

Algorithm 7: [11]

The algorithm described in [11] is applicable to a multi-decision making, multi-level, non-linear problem. The fact that the duality gap for the follower's problem is zero for any specific $x \in \mathcal{X}$ plays the main role. Using the dual form it is easy to find an interval in which the optimal value of the follower's objective function lies. Including the penalty parameter K a new problem can be defined as minimization of objective function (2.1a) penalized by duality gap multiplied by parameter K. The main principle is further described on the page 404 in [11].

In a comparison with other methods, the algorithm easily outperformed the Kth-best algorithm but does worse than the Bard's B&B [20] for larger problems. A disadvantage is that the penalty parameter K is chosen by experiments. This drawback can be eliminated by increasing the penalty parameter K by small discrete steps α at every iteration. The algorithm in [6] is using this approach which makes it different from the first algorithm.

2.2.4 Descent Method Algorithms

In the descent methods for finding a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the gradient, one approaches a local maximum of that function. This procedure is then known as gradient ascent. Gradient descent is also known as steepest descent, or the method of steepest descent [43].

Algorithm 8: [36]

The algorithm described in [36] needs F, f, g to be linear functions and the problem is bounded from below on the $\mathcal{M}(x)$. Using the KKT conditions (1.5) one obtains a non-convex problem in general. A new problem is solved by storing indices of λ_i for which a feasible solution satisfy the inequalities in (2.2g) as an equalities. Even if the solution satisfy the inequalities it does not have to be the optimal one if there exist reduced cost

values for which the obtained solutions are less than 0 (property of the Simplex tableau). The algorithm is able to solve the problem to a locally optimal solution but an improved version can reach a globally optimal solution called "strong locally optimal solution". Such solution is reached when there are no neighboring vertices of set Ω which are feasible points and have smaller function values [36]. That means the global solution is not guaranteed. An example can be found on the page 383 in [36] and steps of the algorithms can be found on the page 381. No comparison with other methods was made.

2.2.5 Trust Region Algorithms

The idea in trust region methods is that the algorithm approximates only a certain region (the so-called trust region) of the objective function with a model function. When an adequate model of the objective function is found within the trust region then the region is expanded. Conversely, if the approximation is poor then the region is contracted. Trust region methods are also known as restricted step methods, [44]. A book about the trust-region methods is [32].

Algorithm 9: [7]

The principle of the algorithm in [7] is the computation of the affine approximation of the original bilevel model (2.1) obtained by replacing f and F by their respective first-order approximations $\overline{f}$ and $\overline{F}$ around the current iterate $(\overline{x},\overline{y})$. Then the upper level decision vector x is forced to lie within some distance ϵ (the radius of the trust region) of the current iterate. Distance ϵ is changed every iteration according to a ratio, which represents how good the trust region model approximates the original model. If the ratio is big ϵ increases, for the ratio in some ranges ϵ remains and for a small ratio ϵ decreases.
The algorithm behaves like a descent algorithm by construction, but by using the KKT conditions (1.5) one can obtain an equivalent primal-dual one-level program with the big M formulation constraints (section 2.1.2). In some occasions a globally optimal solution can be found by implicit enumeration techniques (e.g. B&B) and this makes the algorithm

"semi-global".

A difference between this algorithm and the one in [27] is that the latter one use a penalty function defined from the duality gap in the objective function after reformulation of the problem by the KKT conditions.

2.2.6 Explicit Solution Algorithms

This method belongs to those where a multi-parametric approach is used. The main idea is to first solve the follower's subproblem for every possible value within the follower's feasible set $\Omega(x)$. That means to explicitly compute a parametrized solution $y(x)$ for the follower's subproblem which leads to a piecewise-affine description of the optimizer. After plugging this expression into the leader's subproblem a mixed integer quadratic problem arises.

Algorithm 10: [8]

With the algorithms described in [8], linear BLPP (1.6), quadratic BLPP (1.15), BLPP with a mixed form of the objective functions (LP| QP) and mixed integer BLPP, also with disturbances can be solved. In all the cases linear constraints are considered. The main idea is to divide the follower's feasible set $\Omega(x)$ into K rational follower's reaction sets $\mathcal{M}(x)$, and search for the global optimum of a simple quadratic (or linear) programming problem in each area, [8]. It can be done by considering the optimizer of the leader's subproblem as a parameter in the follower's subproblem. Then the follower's subproblem is recast into the mp-QP form which is solved by the algorithm described on the page 621 in [8] (mp-QP algorithm). Similarly an mp-LP algorithm is used to solve follower's subproblems mp-LP form. That is, problem (2.1d) - (2.1f) is replaced by a piecewise-affine linear functions define over K regions due to the problem. When mp-QP or mp-LP problem is formulated, the algorithm continues in steps describe on the page 614 to finish computations and reach the globally optimal solution.

Numerical results of the linear BLPP were compared with ones presented in [19]. For quadratic BLPP results were compared with those in [30] and for a mixed integer BLPP

comparison was performed by those in [46]. In all cases results were the same but comparison in efficiency of the algorithms or solving time was not done.

2.2.7 DC Programming Algorithms

DC programming is an optimization problem which deals with minimizing of a difference between convex functions. This approach can be used in non-convex optimization, which makes it very important and interesting.

Algorithm 12: [15]

Problem adequate for the algorithm in [15] is transformed into DC programming problem, when the follower's problem is replaced by its primal-dual optimality constraints (KKT conditions). In this case the objective function is represented by the linear function where the penalty function is included. This function is then decompose into the difference of two convex functions. A heuristic algorithm for solution of the newly defined problem can be found on the page 219 in [15].
In [29] one can find algorithms for solving DC programming:

- classic B&B on the page 19
- combination of B&B scheme and Outer-Approximation Method on the page 27
- classic Cutting plane algorithm on the page 34

Numerical experiments were not performed for any of the proposed algorithms.

2.2.8 Genetic Algorithms

A genetic algorithm is a search technique used to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics and they are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover, [45].

Algorithm 13: [33]

The algorithm described in [33] combines the underlying structure of genetic algorithms with some kind of extreme point enumeration. The main idea is to associate chromosomes with extreme points of the polyhedron S which solves the BLPP. This algorithm provides optimal or near-optimal solutions. The fitness of a chromosome will evaluate its quality, penalizing the fact that the associated extreme point is not in an inducible region. After constructing the first population of chromosomes, the genetic algorithm proceeds by performing crossover, mutation, evaluation and selection, until the stopping condition is met. In each generation a population of extreme points of S in maintained [33]. The algorithm can be found on page 17 in [33].

Numerically, the Algorithm 13 was compared with the Kth-best algorithm and with a different genetic algorithm proposed by Hajezi [28]. The algorithm outperformed the Kth-best algorithm in terms of CPU time, size of the computed problems and number of the reached best solutions.

The results from the genetic algorithm of Hajezi and Algorithm 13 were the same. Both algorithms were able to compute large-size problems. However, Hajezi's algorithm needed twice as much computation time to solve the same problem.

2.3 Implemented Algorithms

This section is oriented on a description of implemented algorithms and how they work. These algorithms guarantee a globally optimal solution.

In this section, the form in (2.4) is restricted to problem in a form

$$\min_{x} \ \frac{1}{2}x^T Q x + c^T x + d^T z \tag{2.6a}$$

$$\text{s.t.} \ \ A_{\mathcal{I}} x - E_{\mathcal{I}} z - b_{\mathcal{I}} \leq 0 \tag{2.6b}$$

$$\min_{z} \ \frac{1}{2}z^T H z + e^T z + f^T x \tag{2.6c}$$

$$\text{s.t.} \ \ F_{\mathcal{I}} z - G_{\mathcal{I}} x - h_{\mathcal{I}} \leq 0 \tag{2.6d}$$

All the algorithms rely on the KKT conditions (1.5) for the follower's subproblem, but they are addressed in different ways. To solve this problem is hard, because the convex follower's problem becomes non-convex in the complementarity conditions. As it was mentioned before, in the complementarity constraint (2.2e) only one element can be equal to zero (multiplier or corresponding constraint) for each constraint.

2.3.1 Direct MIQP Approach

This approach is called "direct", because the complementarity conditions in the follower's KKT conditions are "directly" modeled using binary variables (big-M formulation). Better to say, each equation for complementarity constraints (2.2e) is modeled separately as λ_i or $g_i(x)$ (noted as $slack(i)$ in (2.3.1)) is equal to zero

$$i = 1 : \text{length}(h) \tag{2.7}$$

$$KKT = [KKT, ((lambda(i) == 0)|(slack(i) == 0))]; \tag{2.8}$$

$$\text{end} \tag{2.9}$$

It is easy to define these conditions but harder to compute. The way how the algorithm deals with this problem is by introducing one binary variable for each complementarity constraint. This approach is called a big-M formulation (Section 2.1.2). When the so-called IF-THEN-ELSE relation is defined for each line of the complementarity conditions in (2.8) and MILP or MIQP problem is obtained. Formulation of these problems is defined in the Section 1.6 and 1.7. Typical procedure to solve mixed integer problem and find and optimal solution is branch-and-bound. This algorithm also requires to set boundaries of minimal and maximal reachable value on the variables x and y and also a maximal value for dual variable λ must be given.

$$KKT = [KKT, lambda <= 100, -100 <= [x; y] <= 100]; \tag{2.10}$$

For the code see file with name bil_kkt_direct.m on CD. If the objective function in (2.6a) is non-convex, the direct MIQP approach can not be immediately used since MIQP schemes

require a convex objective function However, this can be achieved by using the method described in [24]. As a result, problems where the leader's objective function is a difference of quadratic function can be used.

2.3.2 Global Nonlinear Solver

In this algorithm the global "BMIBNB" solver is used to solve BLPP reformulated by the KKT conditions. This solver is an implementation of a standard branch-and-bound (B&B) algorithm for non-convex problems, based on linear programming relaxation and convex envelope approximations, [13]. Solving non-convex problem by spatial B&B rely on some default steps which will be describe in details.

The first is, for each node in a search tree, a nonlinear solver (e.g. SNOPT or fmincon) is used to compute a feasible and locally optimal solution. This solution represents an upper bound on the achievable objective function value. In the second step, for each node, a convex relaxation of the problem (2.2) is derived. In this case "Linear relaxation for bilevel and quadratic problems" is used. That means, every bilinear and quadratic term is replaced with a newly introduced linear variable. Based on the bounds on the variables in original terms, constraints on the new variables must be defined with the aim to make the relaxation as tight as possible. Based on [12] quadratic terms x^2 and y^2 are replaced by linear variables w_x and w_y, respectively. As an example, every bilinear term xy is replaced by a new linear variable w_{xy} . Then with the given variables x,y and their upper and lower bounds x_U,y_U and x_L,y_L, respectively, the following inequalities are added to the original constraints

$$w_{xy} \geq xy_L - x_L y_L + x_L y \Leftrightarrow (x - x_L)(y - y_L) \geq 0 \tag{2.11a}$$

$$w_{xy} \leq xy_U - x_L y_U + x_L y \Leftrightarrow (x - x_L)(y_U - y) \geq 0 \tag{2.11b}$$

$$w_{xy} \leq xy_L - x_U y_L + x_U y \Leftrightarrow (x_U - x)(y - y_L) \geq 0 \tag{2.11c}$$

$$w_{xy} \geq xy_U - x_U y_U + x_U y \Leftrightarrow (x_U - x)(y_U - y) \geq 0 \tag{2.11d}$$

This procedure is made for every nonlinear term in the problem. The resulting convex optimization problem (a linear program) is solved and the solution represents a lower bound

on the achievable objective. Using the obtained lower and upper bounds, a standard B&B algorithm is used to select a branch variable, branch, prune and finally navigate the global search. For the code see file bil_kkt_bmibnb.m on CD.

2.3.3 Multi-Parametric Programming

In this algorithm, a multi-parametric programming approach is used. To begin with, the follower's subproblem is parametrized by the leader's variable x and is solved explicitly which results into a parametrized solution $y(x)$. During the procedure the follower's feasible set is further restricted into the K small subsets where each of them is described by linear function

$$y_k(x) = m_k + n_k x; \quad H_k x \leq h_k, \quad k = 1, 2, \ldots, K \tag{2.12}$$

where $H_k \in \mathbb{R}^{m \times n}$ and $h_k \in \mathbb{R}^{m \times 1}$ are matrices which describe the *follower's* k^{th} *feasible subset* and m_k,$n_k \in \mathbb{R}$ are coefficients of a k^{th} piecewise affine function. To limit the set where the parametric solution has been computed, bounds on the vector of parametric variables x are added. The parametric solution of the follower's subproblem can be inserted into the leader's subproblem using a big-M formulation (section 2.1.2). Every subset is given by big value of a constant M and corresponding binary variable δ_i, $i = 1, \ldots, K$. To be sure that only one subset is an active condition $\sum_{i=1}^{k} \delta_i = 1$ is added to the problem. Then the algorithm loops through all subsets and solves the problem with corresponding inner optimal parametrization. For the code see file bil_explicit.m on CD.

2.3.4 Branch-and-Cut

This algorithm only applies to BLPP with the convex quadratic or linear follower's subproblem. Moreover, there are no convexity restrictions on the leader's subproblem. The algorithm is based on a B&B procedure [22] where the follower's subproblem is replaced by the KKT conditions.In contrast to the direct MIQP approach in section 2.3.1 branching is done directly on the non-convex complementarity conditions, as a result we avoid the use of problematic big-M formulation. The selection of the outer solver depends on the type

of upper problem and on if we require a globally optimal solution or not. When locally optimal solution for a non-convex problem is enough e.g. the "fmincon" solver can be used, while for a globally optimal solution "bmibnb" solver is required. Then procedure of the convex relaxation described in section (2.3.2) is applied.

The algorithm can be improved by adding disjunctive cuts in root node. The aim is to find an inequality which is already included in the problem and in the same time is not satisfied with in the current convex relaxation. Such inequality is called a *violated cut* which is added to the original constraints. A reason for generating these cutting planes is to remove an optimal "fractional" solution of the problem after the convex relaxation. The main points in the generation of the cutting planes for any 0-1 problem are:

- disjunction - division of inequality conditions into two parts with the same matrices, where one creates plane for the lower value of currently computed value $\lfloor x \rfloor$ and the second one for the upper value $\lceil x \rceil$ by the evaluation of the inequalities
- creation of the convex hull of the union of disjuncted planes
- cut off non-integer solution by facet of the convex hull

When there does not exist more violated cuts, B&B method can proceed. In this algorithm the number of added cuts can be defined by user.

Chapter 3

Comparison of the Algorithms in Numerical Experiments

This chapter focuses on comparison of the algorithms in numerical experiments in terms of solution quality, numerical stability, computational performance, etc. All the results which will be presented in this chapter were computed on computers with 32-bit Intel Xeon CPU 2.80GHz and 32-bit Intel Pentium CPU 3.20GHz. Softwares used for computations were MATLAB 7.10, Yalmip R20100507 and solvers CPLEX 11.1 and BPMPD 2.21. Presented algorithms will be given by names as follows, Algorithm A is described in Section 2.3.1, Algorithm B is one described in Section 2.3.2, Algorithm C is described in Section 2.3.3 and the last one Algorithm D in Section 2.3.4. Algorithm E is a special case of Algorithm A with difference of QP functions

3.1 Convex Bilevel Problems

The focus in this section is computational performance. Each implemented algorithm were run on the same set of 100 convex test problems. This set consist of 4 subsets of 25 problems with different size. A reason for this was to investigate if all the algorithms attain the same results with the same message (exit-flag) from solvers. Also comparison of a time necessary for computations was made.

3.1.1 Generation of Test Problems

The problems used for computations were generated as problems in the form in (2.6). Random matrices Q and H in (2.6a) and (2.6c), respectively, were generated by the command **randn**(n, n). To ensure positivity of the eigenvalues, the matrix is squared and to avoid zeros in the eigenvalues a small value is added to each of the elements on a diagonal of the newly obtained matrix. The result is a matrix with strictly positive eigenvalues. This is, Q and H were computed as

$$Q = \text{randn}(n, n); Q = Q * Q' + 0.1 * \text{eye}(n); \tag{3.1}$$

$$H = \text{randn}(m, m); H = H * H' + 0.1 * \text{eye}(m); \tag{3.2}$$

where n and m represents number of the x and y variables, respectively. The linear terms in the objective functions of the problem (2.6) were generated as

$$c = \text{randn}(n, 1)); d = \text{randn}(m, 1); \tag{3.3}$$

$$e = \text{randn}(m, 1)); f = \text{randn}(n, 1); \tag{3.4}$$

For the concrete steps see file gener.m on CD.

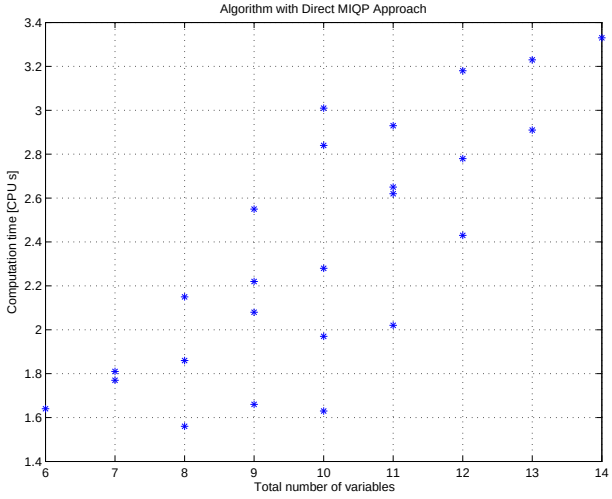
3.1.2 Computation time

Each of the algorithms use different methods to solve the problem. Depending on the method, different computation time is necessary to perform the computations which also depend on the number of variables presented in the problem. As it can be seen from the Figures 3.1(a),3.2(a) and 3.3(a) the computation time is increasing with every new introduced variable. The largest time was measured for the BMIBNB solver algorithm. The performance of this algorithm is similar with the next two algorithms only in problems with small total number of variables. The computation time spent in the Algorithm B for

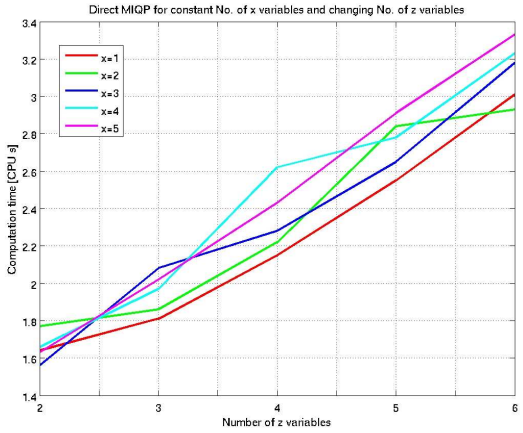
problems with a total number of 7 variables is approximately the same as the computation time of the Algorithm A for 14 variables. The computation time for the Algorithm D for a problem consisting of 14 variables is almost the same as a time for the algorithm with BMIBNB solver to solve a problem with 6 variables. The algorithms depicted in the Figures 3.1(a) and 3.3(a) are of the same efficiency due to the computation time.

Now, look closer on the graphs 3.2(a) and 3.2(b). In 3.2(a), fix the total number of variables to for example 10. Then one can find 5 dots representing different computation times. This is a result of that the generated problems consist of different combinations of the follower's and leader's number of variables which sum up to the same total number of variables. In Fig. 3.2(b) the connection between the computation time and number of variables y in the follower's subproblem is depicted. Each line there represents how the computation time changes when the number of the leader's variables is fixed and the number of the follower's variables is changing. As it can be seen from the Figures 3.1(b) - 3.3(b) the computation time is an increasing with the increasing number of the follower's variables. As mentioned before, all three algorithms are similar in terms of computation time for small problems. This is confirmed by Fig. 3.4(d) where the normalized times are the same. By the term normalized it is meant the obtained times for one combination of variables are divided by their maximal value. In this way it is easier to compare times when there are large differences between them. But it doesn't mean that the original times are the same. Normalized time represents computating speed in comparison to the slowest algorithm From the rest of the figures in Fig. 3.4 it can be seen that the computation times depend on the ratio of x and y variables. The Algorithm A (Direct MIQP in the graph) is better when the number of the leader's variables is larger than the number of the follower's variables. This is not true for the Algorithm with nonlinear solver for complementarity constraints (BMIBNB solver in the graph) where the ratio between the number of variables does not effect time needed for computations. It is changing without clear or dependences. The Algorithm D (Branch-and-Cut in the graph) shows more-or-less the same computation time for each ratio. Description of the Algorithm C is missing, because the computation time is not comparable with the other algorithms. The algorithm spends a large amount

of time when computing the explicit solution. The explicit solution for the follower's subproblem was computed in the box $-100 \leq x \leq 100$. This means, the algorithm is comparable in term of the computation time only for very small problems.



(a) Average computation time for the Algorithm A depending on the total number of variables in the problem

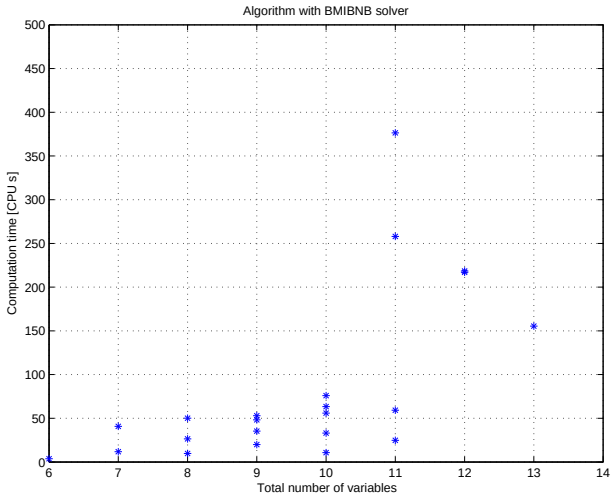


(b) Average computation time for the Algorithm A where is fixed No. of x variables and changing No. of y variables

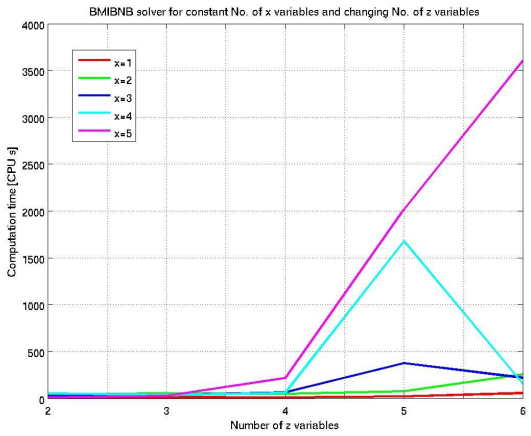
Figure 3.1: Computation time depending on the number of variables

3.1.3 Choice of Solvers in Branch-and-Cut

In this part the main focus will be given on the behavior of the Algorithm D when two different solvers are used to solve the subproblems. The question is, what is the influence of the method used by the solver on the computation time and number of nodes of the tree which is necessary to explore. The used solvers were CPLEX and BPMPD. CPLEX can be used to solve LP, QP, MILP and MIQP problems while BPMPB represents an interior point solver. The computational times are presented in Fig. 3.5(a). In most cases the computation times are the same, but in some the BPMPD solver performs better. It can be seen from Fig. 3.5(b) that the normalized computation time is the same for small problems. For the huger ones (see Fig. 3.5(d)) the computation times depend on the number of variables in the leader's and follower's subproblem. To conclude, it is better to

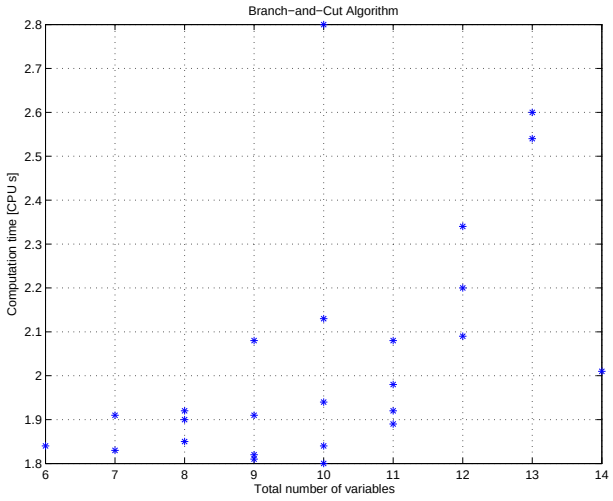


(a) Average computation time for the Algorithm B depending on the total number of variables in the problem

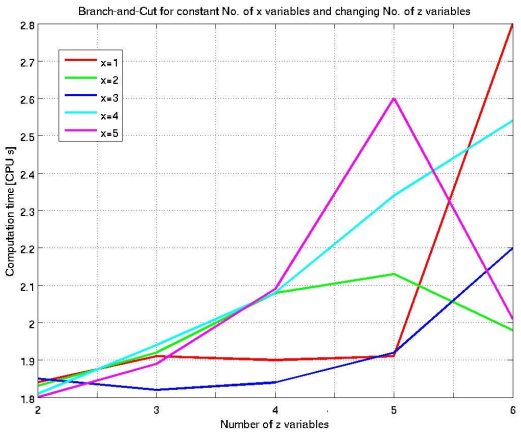


(b) Average computation time for the Algorithm B where is fixed No. of x variables and changing No. of y variables

Figure 3.2: Computation time depending on the number of variables

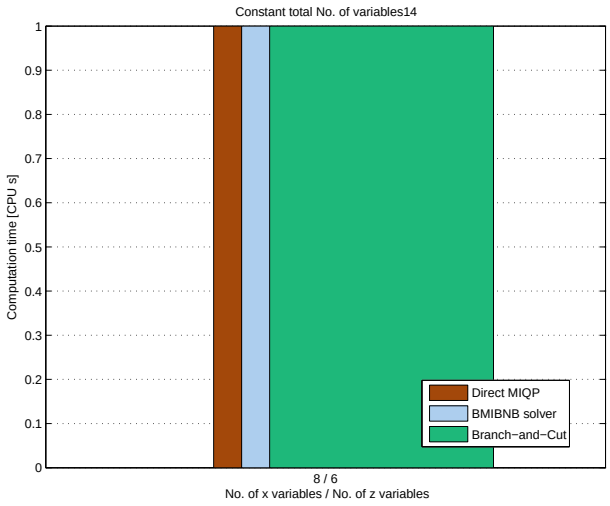


(a) Average computation time for the Algorithm D depending on the total number of variables in the problem

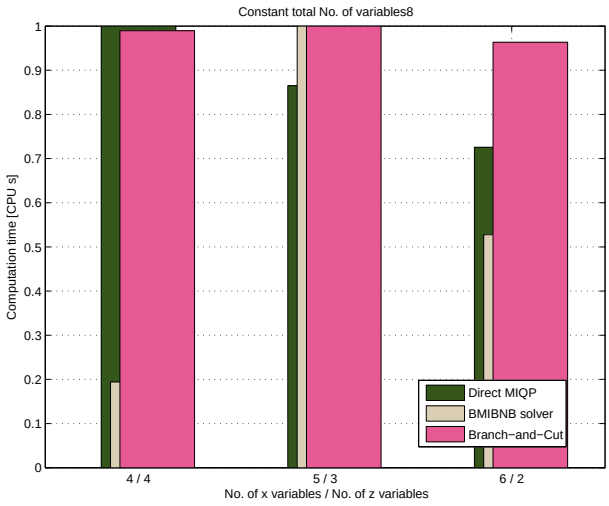


(b) Average computation time for the Algorithm D where is fixed No. of x variables and changing No. of y variables

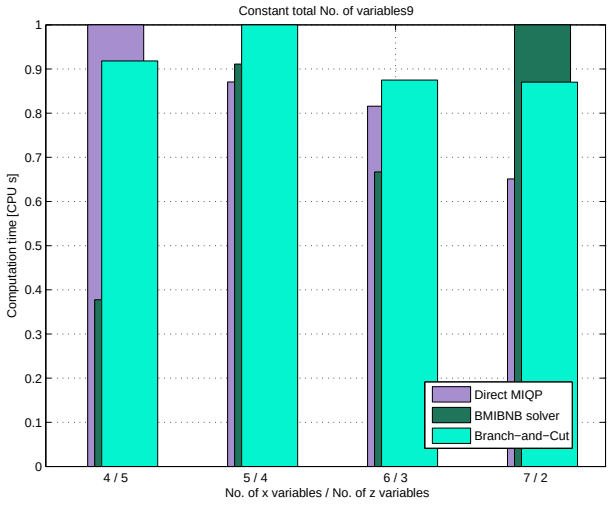
Figure 3.3: Computation time depending on the number of variables



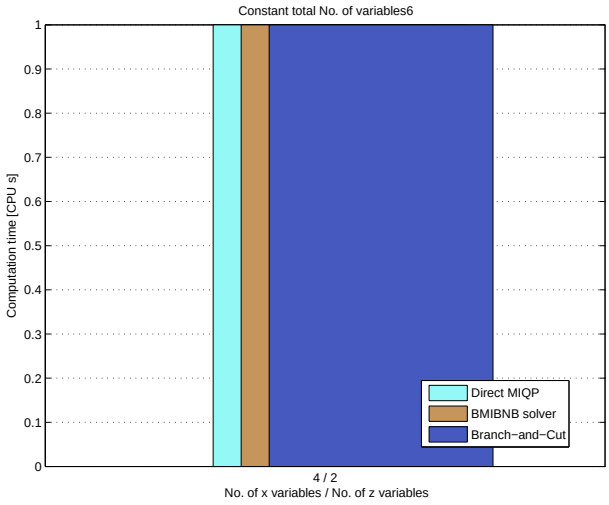
(a) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. of variables 14



(c) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. 8 variables



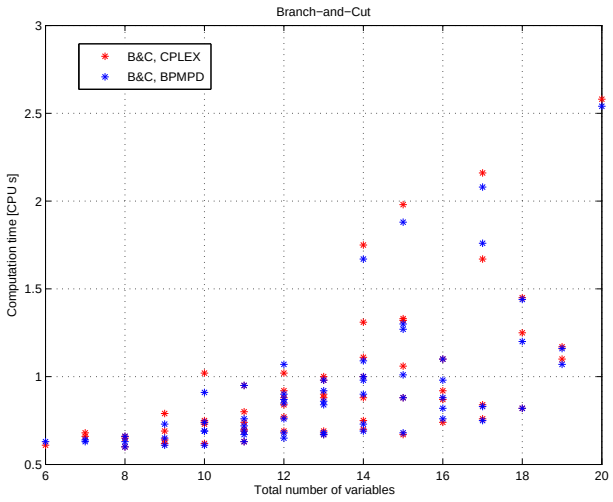
(b) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. of variables 9



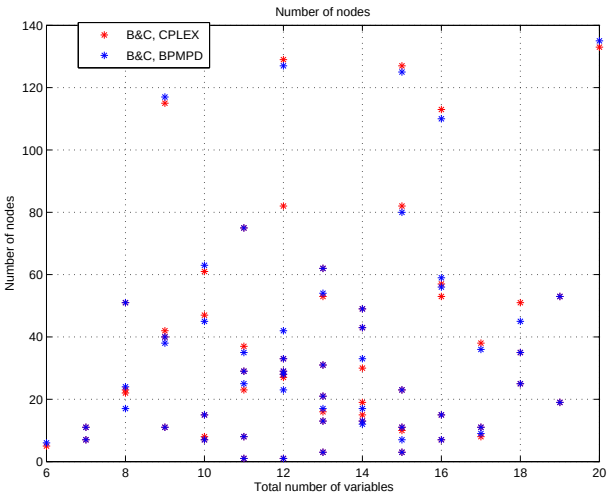
(d) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. 6 variables

Figure 3.4: Computation time for convex problems depending on the ratio of the variables

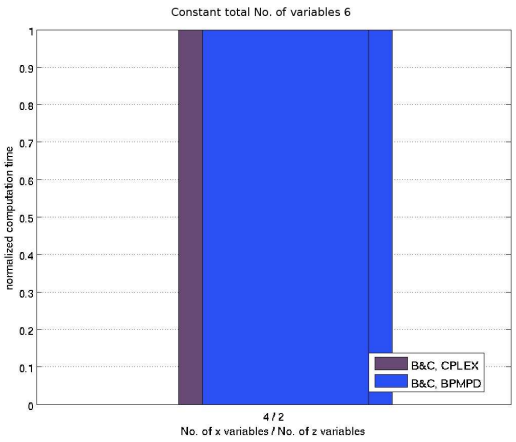
have less leader's than follower's variables in the problem and as it was mentioned before BPMPD solver performs slightly better than CPLEX, which can be clearly seen in the graphs.



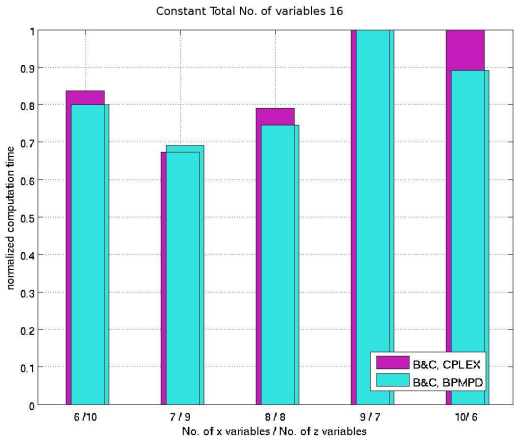
(a) Computation time for different test problems



(c) Number of nodes for investigation in different test problems



(b) Comparison of computation time based on the rate of variables x/y for total number 6



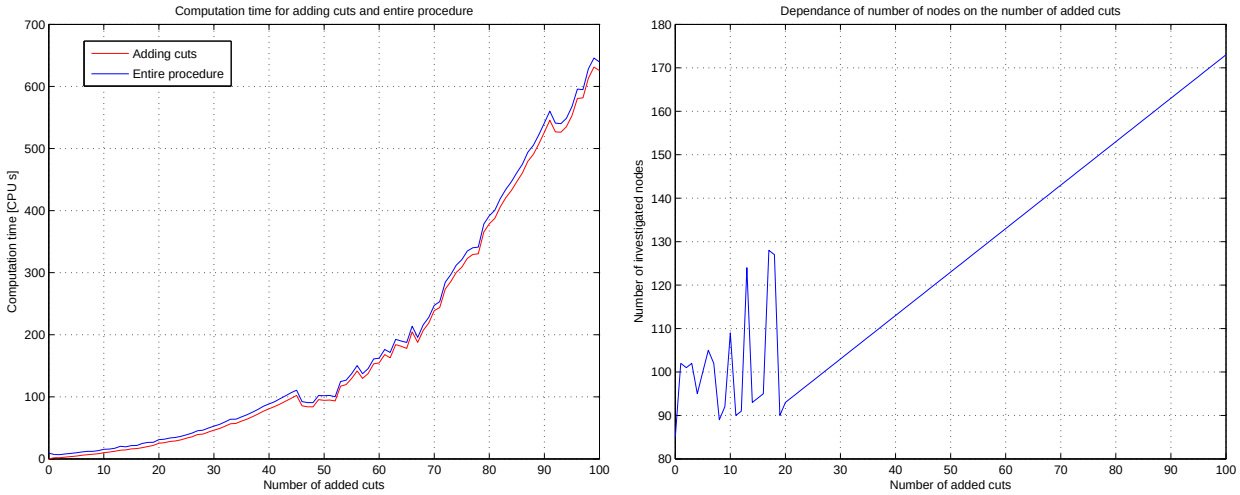
(d) Comparison of computation time based on the rate of variables x/y for total number 16

Figure 3.5: Efficiency of solvers used in Algorithm D

3.1.4 Branch-and-Cut and Cuts

By adding cuts to the root node, one can get rid of needless solutions at the beginning.The purpose is to solve the problem quicker and also explore less nodes in a tree. Experimental results depicted in Fig. 3.6 were obtained for the problem with ten x and ten y variables. In Fig. 3.6(b) it can be seen that the number of the investigated nodes was changing rapidly until the number of additional cuts was equal to total number of variables, i.e., 20. Thereafter, the number of nodes was changing linearly. Note that, according to Fig. 3.6(b) the number of explored nodes was always bigger than in the case without additional cuts. Influence of the cuts is notable in term of the computation time which is increasing with the number of added cuts. A large amount of time is spent on adding cuts not on the B&B procedure. Graph 3.6(a) shows that difference between time for adding cuts and entire algorithm's procedure is very small.

Collecting information about the number of nodes and the computation time results in knowledge, that it is more efficient to use Algorithm D without additional cuts.



(a) Computation time

(b) Number of investigated nodes

Figure 3.6: Number of investigated nodes and computation time depending on the number of added cuts

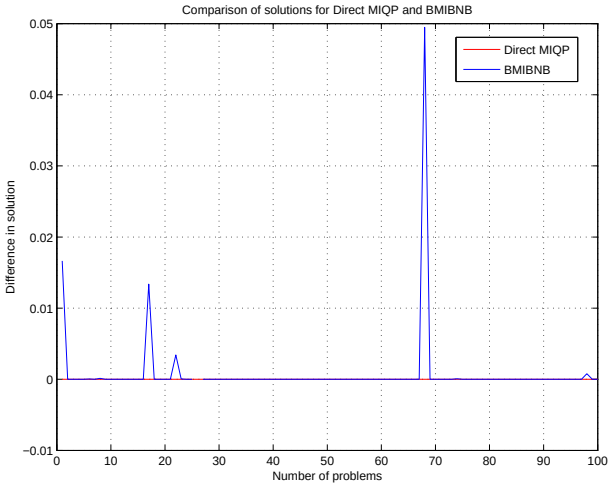
3.1.5 Reliability of the Algorithms

The algorithms were applied to the problems with different number of the leader's and follower's variables. The number of variables varies from 4 to 8 and from 2 to 6 for the leader's and follower's subproblem, respectively. The results are expected to be the same and globally optimal for the all algorithms.

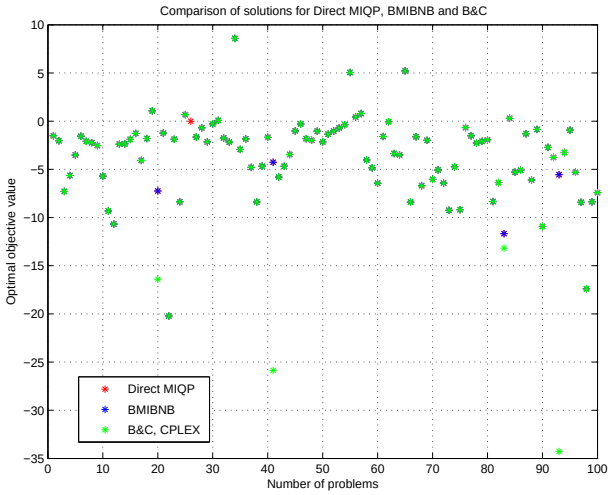
The comparison was based on the optimal value of the leader's objective function. Values from the Algorithm A were chosen as a reference (f_{ref}). These values were then subtracted from the results of the other algorithms (f_{test}) for the same problems. In a mathematical representation $f_{test} - f_{ref}$. Zero difference then means the solutions are the same. If a difference is positive, the solution from the algorithm is only a local optimum because the objective function value is bigger that one from reference algorithm. If a difference is negative, this predicates the result from reference alhorithm is not globally optimal. This means that a lower objective function value for the minimization problem was obtained by the second algorithm and not by a reference one. The gap between the upper and lower bound plays important role in the algorithms with B&B procedure. In Fig. 3.7(a) can be seen that same values are not reached every time .

Consider the problem in Fig. 3.7(a), where the largest difference is 0.05. This means, that either, Algorithm A or Algorithm B may reached the solution which does not satisfy follower's or leader's conditions or a locally optimal solution. From Fig. 3.7(d) it follows that Algorithm D reached a solution of the same as the Algorithm A due to zero difference, Algorithm D with a zero gap between upper and lower bound. For the Algorithm B, 0.64% gap between bounds indicates a locally optimal solution as a result. To determine whether the solution is feasible, the follower's and leader's constraints are tested for feasibility. Constraints for this problem were satisfied that results in the feasible, but a locally optimal solution for Algorithm B. Since the difference is positive, the solution is a local optimum.

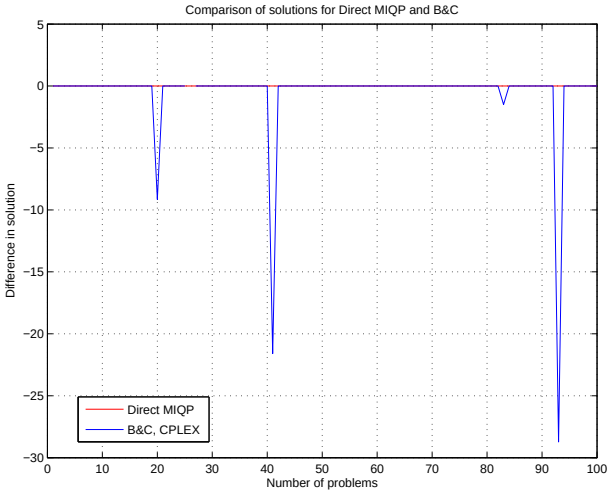
Regarding the reliability of the algorithms, different optimal values were reached only in 9 tested problems out of 100. Most of the times all three algorithms reach the same optimal obejctive value. When one algorithm obtained different solution and difference is positive



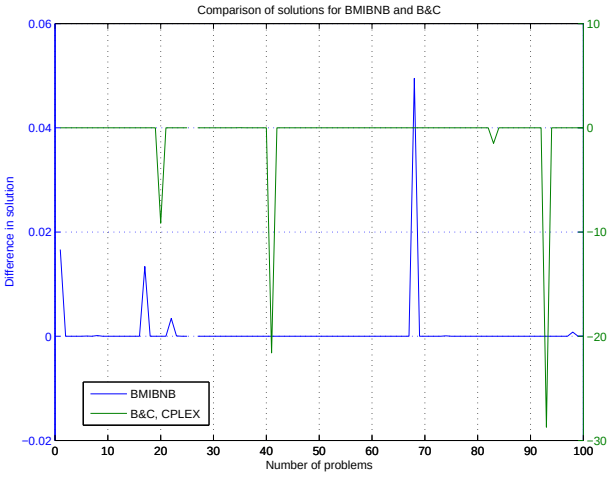
(a) Comparison of solutions for Algorithms A and B



(b) Comparison of solutions for Algorithms A, B and D



(c) Comparison of solutions for Algorithms A and D



(d) Comparison of solutions for Algorithms B and D

Figure 3.7: Comparison of solutions

the remaining two reched the global optimum. If a difference between the objective values is negative the algorithm perform better and thus remaining two reached only a local optimum. These remarks hold only when the solutions satisfay both, the follower's and leader's constraints. In case of Algorithm B if difference appeard it was of possitive value, that is only the locally optimal solution was reached. The Algorithm D performed better because the globally optimal solution was reached since the negative value of the difference (see Fig. 3.7(c)). In fact, the best performance was shown by the Algorithm D, which reached a global optimum for each problem. The corresponding objective function values are depicted in Fig. 3.7(b) where each point represent the objective value for one of the problems. The second most reliable approach was by the Algorithm A and finally, the worst performed the Algorithm B. Regarding computation time, the best was Algorithm D followed by Algorithm A. The Algorithm B gave the largest computation time. Here it should be noted that the Algorithm D was under development at the time of this project. Many bugs were found and corrected and also time of performance improved significantly. The Algorithm C was able to reach the same results only for a small subset of problems i.e., the algorithm failed for problems with more than 10 variables. This means, the algorithm can not be used for solving large problems. It should be noted though, that in the explicit approach might be possible to improve the performance by solving the follower's subproblem for a smaller box than $-100 \leq x \leq$ which is more carefully selected for the given problem.

3.2 Non-convex Bilevel Problems

This section gives an overview in the performance of the algorithms. Each implemented algorithm was run on the same set of 100 non-convex test problems with the aim to investigate the computational times and the reliability of the different algorithms.

3.2.1 Generation of Test Problems

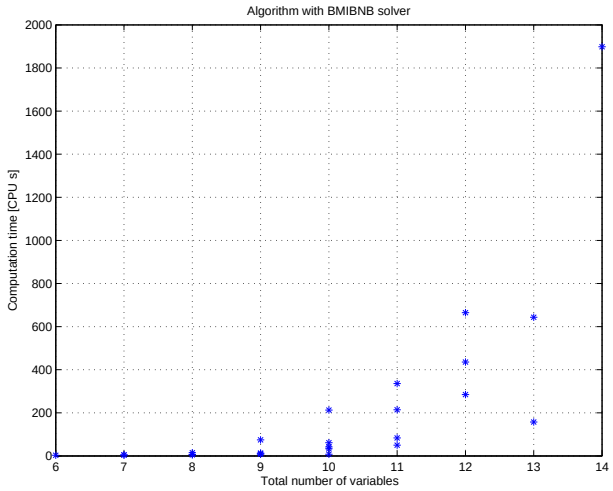
Test problems have been generated with a nonconvex leader's objective function. More exactly the steps in the program for generation are the same as in the Section 3.1.1 except for the generation of the Q matrix (see file gener_nonconvex on CD). Nonconvexity means that at least one eigenvalue is negative. This was achieved by inserting two negative values in the diagonal of the Q matrix.

$$Q(2,2) = -Q(2,2);\ Q(n,n) = -Q(n,n); \tag{3.5}$$

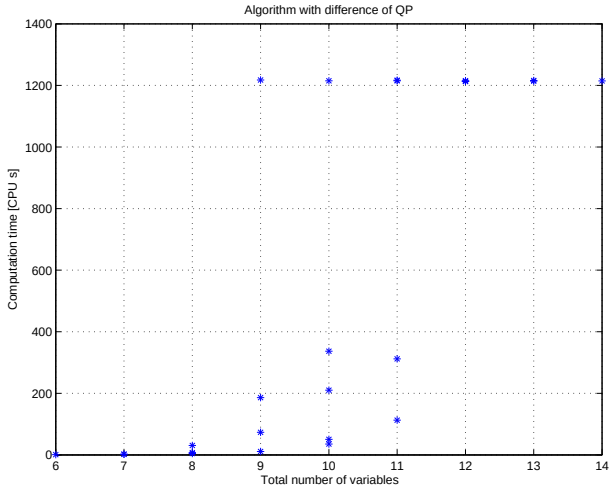
where n represents number of x variables.

3.2.2 Computation time

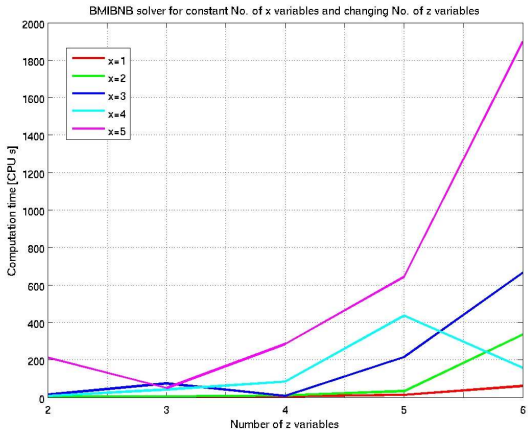
Since the non-convex problems are harder to compute, different algorithms than in the convex case are used for solving them and larger computation times are to be expected. This section describes the performance of the Algorithm B and the Algorithm E. The computation times are approximately the same as it can be seen from the Figs. 3.8(a) and 3.8(b) but Algorithm B perform a bit faster. Fig. 3.8(b) shows many dots on the time 1200 [CPU s] which was chosen as the maximum computation time for CPLEX. Each dot in the vertical line belongs to problem with the same total number of the variables but different ratio between the leader's and follower's variables. Changing the ratio of these variables results in different computation times as depicted in Fig. 3.9. Both algorithms give the same computation time for small problems. For larger problems with more than 12 variables the normalized times are the same because both algorithms reached their maximum computation times. The computation time is decreasing in the smaller problems for the Algorithm E with increasing number of leader's variables (see Fig. 3.9(b)).
The influence of the number of the follower's variables for the fixed number of the leader's variables is depicted in the Figures 3.8(c) and 3.8(d).



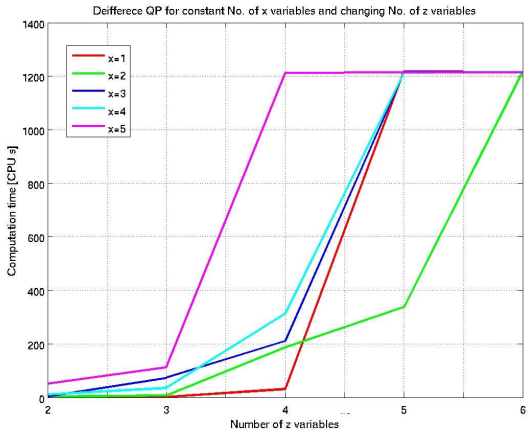
(a) Computation time for the Algorithm A depending on the total number of variables in the problem



(b) Computation time for the Algorithm B depending on the total number of variables in the problem

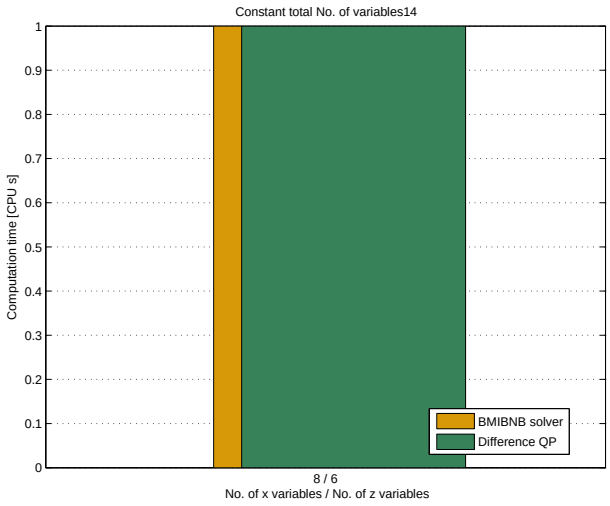


(c) Computation time for the Algorithm A where is fixed No. of x variables and changing No. of y variables

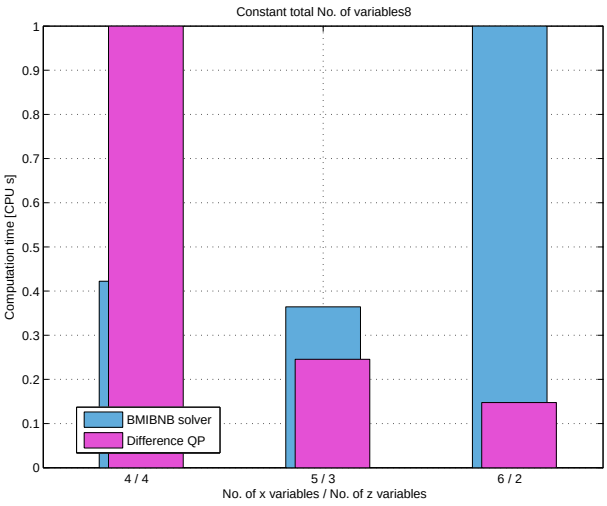


(d) Computation time for the Algorithm where is fixed No. of x variables and changing No. of y variables

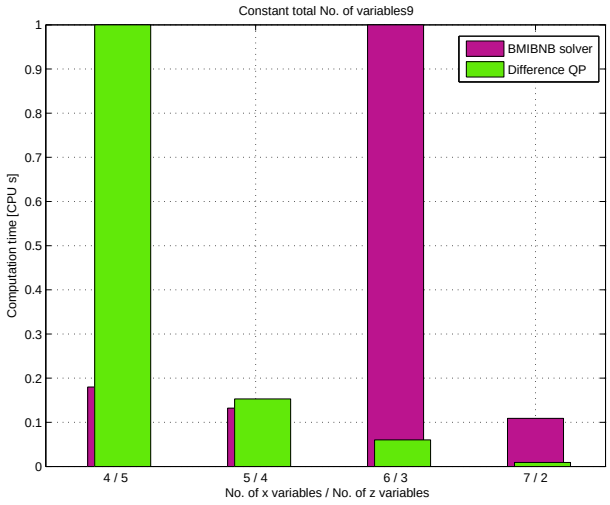
Figure 3.8: Computation time for non-convex problems depending on the number of variables



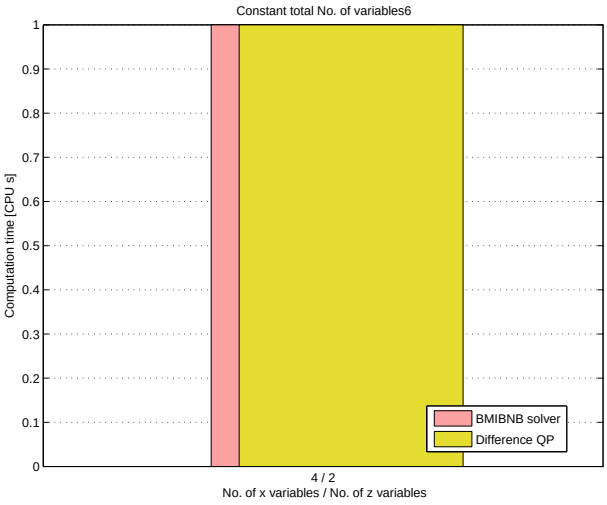
(a) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. 14 variables



(c) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. 8 variables



(b) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. 9 variables



(d) Comparison of algorithms in term of computation time based on the ratio of variables x/y for total No. 6 variables

Figure 3.9: Computation time for convex problems depending on the ratio of the variables

3.2.3 Reliability of the Algorithms

The gap between upper and lower bound plays an important role for reliability of algorithms. When the gap is 0% a solution proven to be globally optimal has been found. However, for the larger values of this gap, the solution can represent only a locally optimal solution. This information is useful to confirm the globality of the solutions for 100 non-convex random test problems. The comparison is based on the optimal objective function values obtained from the Algorithm B and Algorithm E. The first objective function value was set as a reference and its values were subtracted from the second objective value for each problem. This results in the difference of the objective function values depicted in Fig. 3.10. A free space in the graph represents an infeasible solution due to infeasibility of the original problem. In a place where peaks appear the solutions differ.

We will now look at the influence of the size of the gaps for the outputs from the algorithms in these problems. It can be considered that better solution was reached by the Algorithm B because the gaps are maximally of the size 0.03% which is less then for Algorithm E. This shows that the solution from the Algorithm B is close global.

To conclude, both algorithm attain the same results in 96 problems out of 100. Solution of one problem was infeasible due to the infeasibility of the original problem. The Algorithm B performs better in the remaining three problems because values reached by the Algorithm E (10^9) are too far to be an optimal solution. Since the Algorithm E reached maximamum computation time, then the best computed value is given as a solution which most likely is not optimal.

3.3 MPC Bilevel Test Problem

In this section a comparison of the algorithms on a Model Predictive Control (MPC) problem is performed.

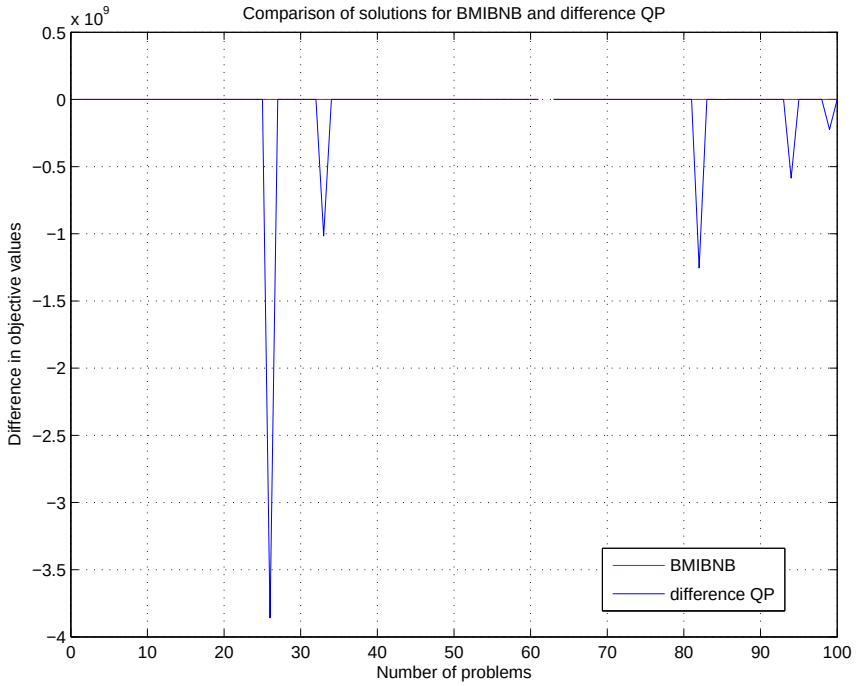


Figure 3.10: Comparison of the solutions for nonconvex problems

3.3.1 Problem Description

Consider the MPC optimization problem in [26] that uses a linear model of the plant and quadratic type of the objective function for the states $x_k \in \mathbb{R}^{4\times 1},\ k = 1, \ldots, N$ and the control inputs $u_k \in \mathbb{R}^{2\times 1}$, $k = 1, \ldots, N-1$.

$$J_N^*(x) = \min \frac{1}{2} x_N^T P x_N + \frac{1}{2} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \tag{3.6a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, \ldots, N-1 \tag{3.6b}$$

$$u_k \in \mathbb{U}, \quad k = 0, 1, \ldots, N-1 \tag{3.6c}$$

$$x_0 = \overline{x} \tag{3.6d}$$

where the constraints in (3.6) arise naturally in practice and $\overline{x}$ is the measured state.

If the system is linear, the problem in (3.6) can be written in the form

$$J_N^*(x) = \min \frac{1}{2}U^T \mathscr{T} U + x^T \mathscr{L} U + \frac{1}{2}\mathscr{M} x \tag{3.7a}$$

$$\text{s.t.} U \in \mathbb{U}^N \tag{3.7b}$$

where U is sequence of inputs over the control horizon defined as $U = (u_0, u_1, \ldots, u_{N-1})$ and constrained to be in the set $\mathbb{U}^N = \mathbb{U} \times \ldots \mathbb{U} \subset \mathbb{R}^{Nm}$ with Hessian matrix $\mathscr{T} \in \mathbb{R}^{Nm \times Nm}$ and matrices $\mathscr{L} \in \mathbb{R}^{n \times Nm}$ and $\mathscr{M} \in \mathbb{R}^{n \times n}$ being easily derived from the problem (3.6). This problem is used to minimize number of iterations in during the optimization procedure. Then bilevel optimization problem appears for the least conservative lower bound on the number on iterations in a form

$$\max \frac{1}{2}U^*(x)^T (LI - \mathscr{T}) U^*(x) \tag{3.8a}$$

$$\text{s.t. } x_{min} \leq x \leq x_{max} \tag{3.8b}$$

$$U^*(x) = \arg \min_{U \in \mathbb{U}^N} J_N(U; x) \tag{3.8c}$$

where $\mathbb{U} = \{u | Hu \leq K\}$ and L and I metrices came from definition of a lower bound [26]. The follower's subproblem is reformulated by the KKT conditions (1.5) into a form

$$\mathscr{T} U^* + \mathscr{L}^T x + \mathscr{H}^T \lambda^* = 0 \tag{3.9a}$$

$$\mathscr{H} U^* \leq \mathscr{K} \tag{3.9b}$$

$$\lambda^* \geq 0 \tag{3.9c}$$

$$\lambda^{*T} (\mathscr{H} U^* - \mathscr{K}) = 0 \tag{3.9d}$$

where $\mathscr{H} = I_{N \times N} \otimes H$ and $\mathscr{K} = \mathbf{1}_{N \times 1} \otimes K$.

However, algorithm are able to solve only minimization problems, the problem (3.8) becomes a noncovnex one-level problem in a form

$$\min \quad -\frac{1}{2}U^*(x)^T(LI - \mathscr{T})U^*(x) \tag{3.10a}$$

$$\text{s.t. } x_{min} \leq x \leq x_{max} \tag{3.10b}$$

$$\mathscr{T}U^* + (L)^T x + \mathscr{H}^T\lambda^* = 0 \tag{3.10c}$$

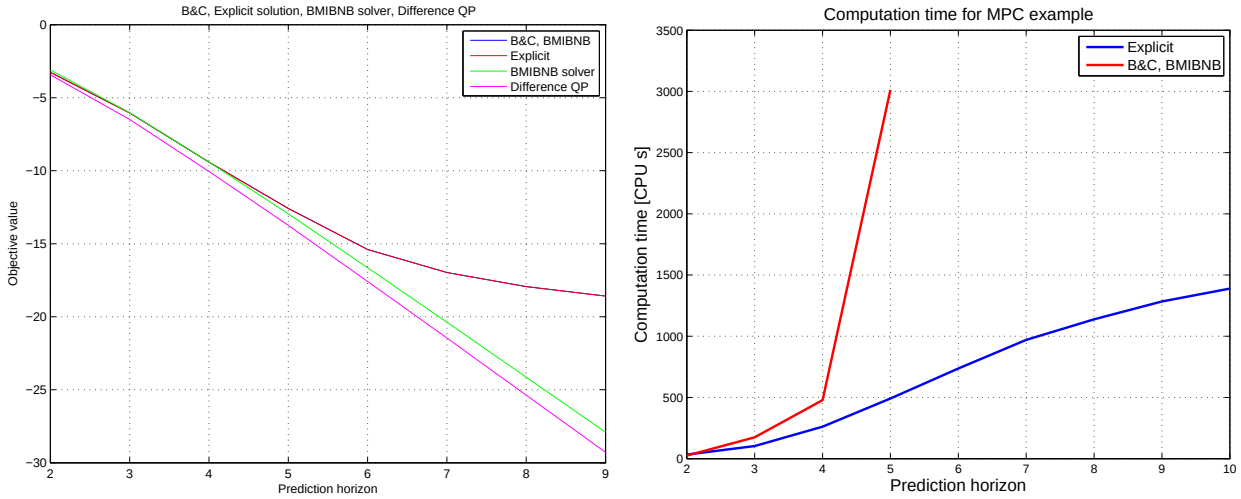
$$\mathscr{H}U^* \leq \mathscr{K} \tag{3.10d}$$

$$\lambda^* \geq 0 \tag{3.10e}$$

$$\lambda^{*T}(\mathscr{H}U^* - \mathscr{K}) = 0 \tag{3.10f}$$

3.3.2 Results

Four algorithms were applied to this problem, concretely Algorithm B, Algorithm D, Algorithm E and Algorithm B used BMIBNB solver.



(a) Computation results for the MPC example

(b) Comparison of solutions for the Algorithms A and D

Figure 3.11: Results for MPC example

Fig. 3.11(a) shows the optimal value of the objective function in (3.10a) as a function of the prediction horizon. As can be seen in the plots Algorithm D with BMIBNB solver (B&C in graph) and Algorithm C (explicit in graph) reached a similar objective function value. However, the Algorithm B (BMIBNB in graph) and Algorithm E (Difference QP

in graph) gave different results. From the Fig. 3.11(a) one could expect that the result for the Algorithm E is the global optimum because it obtained the lowest value. This is not true, because a further investigation of the solution showed that the solution of the follower's problem is not optimal. The problem might be connected with the choice of the big-M values. The Algorithm B neither gave an optimal solution for the follower's subproblem. If the solution for the follower's subproblem is not optimal, then the solution for the entire bilevel problem is infeasible, and hence, better objective function values might be attained.The result is that the best feasible results were obtained by the Algorithm D and Algorithm C and we believe that these are the globally optimal ones. However, the second one only for smaller problems.

Since only two algorithms were able to compute a usful solution, their computation time will be examined. The time used by Algorithm C was increasing slowly with the size of the prediction horizon (see Fig. 3.11(b)). The computation time for ALgorithm D increased dramatically as a length of the prediction horizon grew. In Fig. 3.11(b) only prediction horizon lengths shorter than 5 are considered for the Algorithm D because larger problems became intractable.

From the results above one can conclude that the best algorithm for this non-convex problem is Algorithm C using BMIBNB solver. It reached a globally optimal solution in a reasonable amount of time.

Conclusions

The bilevel optimization problems belong to the class of $\mathcal{NP}$-hard problems,which loosely means that no *polynomial time algorithm* exist for solving it unless $\mathcal{P} = \mathcal{NP}$[17]. They can be described as an optimization problem constrained by another optimization problem is called a bilevel programming problem (BLPP). From the mathematical point of view it is a problem with hierarchical structure where two independent decision-makers appear. Nowadays, this problem becomes useful to define problems in an engineering or automatic control. Unfortunately, these problems are known to often be very hard to solve in practice. The main focus in this thesis was therefore to investigate different algorithms to solve them efficiently. At the beginning the forms of bilevel programming problem were briefly described and a literature overview of the algorithms was made. Afterwards, the algorithms were briefly described with a focus on the type of the solved problems and computation procedure. Regarding the methods, more than 10 groups of algorithms were obtained,[36]:

- Branch-and-bound algorithms
- Cutting plane algorithms
- Barrier and penalty functions algorithms
- Descent algorithms
- Trust region algorithms
- SQP algorithms
- Heuristics algorithms

- Algorithms for solving follower's subproblem explicitly (e.g. mp-LP, mp-QP)
- DC programming
- Genetic algorithms

From these algorithms some promising ones (algorithm with direct MIQP approach, algorithm with global nonlinear solver, branch-and-cut and algorithm with explicit solution) were selected to be more thoroughly investigated in numerical experiments on a large number of randomly generated test problems with the purpose to evaluate the computational performance and reliability. In these experiments, bilevel problems with convex quadratic programming problems at the lower level and convex, as well as non-convex, quadratic programming problems at the upper level were considered.

Firstly the algorithms were run on a set of random convex problems and the results were iinvestigated in terms of computation times and reliability. It was shown that the computation times depend on the distribution of the variables for the follower's and leader's subproblem and on the number of additional cuts in a case of branch and cut. For the convex bilevel problem the algorithm with BMIBNB solver performed worst in terms of computation time and reaching the globally optimal solution. The best was branch-and-cut without adding any cuts. In experiments for non-convex problems, only branch-and-cut, algorithm with explicit solution using BMIBNB solver and algorithm with BMIBNB solver could be used. Also special form of the algorithm with direct MIQP approach can be used, where the objective function is in a form of a non-convex quadratic function. The compared algorithms required nearly the same computation times but they differed in the values of a globally optimal solution. Also in this case branch-and-cut performed best.

Also a test problem from real world were considered. This problem arises in an application of evaluation of the performance of an on-line MPC algorithm. The correct results were reached only by branch-and-cut and algorithm with explicit solution. In a case of algorithm with BMIBNB solver and algorithm with difference of QP function results could not be considered correct because the returned solution was not feasible.

To conclude, the algorithm which performed best in terms of computation time and reliability for both the convex and non-convex problems was branch-and-cut. The algorithm one could also trust to reach a global optimum was algorithm with explicit solution but it required more computation time and could be used only for a subset of small problems.

Resumé

Dvoj-úrovňový (bilevel) optimalizačný problém predstavuje dva optimalizačné problémy, kde premenné v jednom optimalizačnom probléme (problém hornej úrovne) musia byť optimálne v druhom optimalizačnom probléme (problém dolnej úrovne). Tento druh problémov sa môže objaviť ako v procesnom inžinierstve tak aj pri plánovaní v hospodárstve. Nedávne výskumy ukázali, že týmto spôsobom sa dajú definovať a riešiť aj problémy v oblasti automatického riadenia.

Dvoj-úrovňový problém predstavuje dva hierarchicky usporiadané optimalizačné problémy v nasledujúcej forme

$$\min_x \ F(x, y(x)) \tag{3.11a}$$

$$\text{s.t. } G(x, y(x)) \leq 0 \tag{3.11b}$$

$$H(x, y(x)) = 0 \tag{3.11c}$$

$$y(x) = \operatorname*{argmin}_y \ f(x, y) \tag{3.11d}$$

$$\text{s.t. } g(x, y) \leq 0 \tag{3.11e}$$

$$h(x, y) = 0 \tag{3.11f}$$

kde podľa typu účelových funkcií $F(x, y)$ a $f(x, y)$ rozoznávame viacero prípadov. Prvým je prípad, kedy sú obe účelové funkcie lineárne. Vtedy sa hovorí o lineárnom bilevel probléme. Ak jedna s nich nadobúda kvadratickú formu, jedná sa o kvadratický bilevel problém. Pridaním neurčitostí sa z daných typov stáva bilevel problém s neurčitosťami. Nakoniec zahrnutím celých čísel sa dopracujeme k zmiešanému celočíselnému bilevel problému.

Hlavný dôraz v tejto práci bol daný na dvoj-úrovňové problémy, kde v hornom probléme sa

nachádzala konvexná alebo nekonvexná kvadratická účelová funkcia a pre dolný problém to bola iba konvexná kvadratická funkcia. Do úvahy boli brané iba ohraničenia lineárneho tvaru pre obe úrovne. Algoritmy pre tento typ problémov boli rozobrané a porovnané v teoretickej rovine. Väčšina algoritmov bola vhodná na riešenie konvexných problémov a niektoré iba na nekonvexné. Najväčšia množina algoritmov bola založená na metóde vetiev a hraníc. Ďalšie algoritmy boli založené na metódach ako sú metóda rezných nadrovín (Cutting plane method), metóda bariér a pokutových funkcií, gradientové metódy, metóda dovoleného priestoru (Trust region method), heuristické metódy, genetické algoritmy ako aj metódy DC programovania.

Základným princípom riešenia dvoj-úrovňového problému bola reformulácia dolného problému pomocou nutných a postačujúcich podmienok optimality, takzvaných Karush-Kuhn-Tuckerových podmienok. Týmto krokom sa dvoj-úrovňový problém (3.11) zmenil na problém jednej úrovne v tvare

$$\min_{x,y,\lambda,\mu} \quad F(x,y) \tag{3.12a}$$

$$\text{s.t.} \quad G(x,y) \leq 0 \tag{3.12b}$$

$$H(x,y) = 0 \tag{3.12c}$$

$$\nabla_y f(x,y) + \lambda^T \nabla_y g(x,y) + \mu^T \nabla_y h(x,y) = 0 \tag{3.12d}$$

$$\lambda_i g_i(x,y) = 0, \quad i = 1,\ldots,m \tag{3.12e}$$

$$\lambda \geq 0 \tag{3.12f}$$

$$g(x,y) \leq 0 \tag{3.12g}$$

$$h(x,y) = 0 \tag{3.12h}$$

V tomto probléme sa však nachádza nekonvexný člen v rovnici (3.12a). Hovorí o tom, že aby bola splnená daná podmienka, buď $\lambda = 0$ alebo $g(x,y) = 0$. Toto je ošetrené pomocou metódy, ktorá sa nazýva big-M formulácia. Jej princípom je, že každému z členov je pridaná nová skalárna premenná M s veľkou hodnotou a tiež binárna premmená. Binárna premenná hovorí o tom, ktorá z podmienok je v aktuálnom momente aktívna. Takto sa

získa zmiešaný celočíselný problém, ktorý sa môže riešiť napríklad metódou vetiev a hraníc. V práci boli na 100 náhodných testovacích problémoch, konvexných aj nekonvexných, testované 4 algoritmy. Konkrétne, algoritmus s priamym prístupm k riešeniu zmiešaného celočíselného kvadratického problému, algoritmus s globálnym nelineárnym solverom, algoritmus vetiev a rezov a nakoniec algoritmus s multi-parametrickým prístupom. V ďalšom bolo sledované ako sa algoritmy správajú na súbore príkladov z hladiska výpočtoveho času a spoľahlivosti riešenia.

Výsledkom bolo, že vplyv na výpočtový čas má tak celkový počet premenných v dvojúrovňovom probléme ako aj v rozložení daného počtu premmných medzi problém hornej a dolnej úrovne. Pri metóde vetiev a rezov, sa zistilo, že pridanie rezov do hlavného uzlu stromu zvýši čas potrebný na výpočet a nezníži počet uzlov, ktoré je potrebné preskúmať pri dosiahnutí rovnakého výsledku. Najvhodnejšie je teda rešiť problém bez nepridaných rezov. V oboch prípadoch, pre konvexné aj nekonvexné problémy, vyšiel najlepšie z hľadiska výpočtového času ako aj dosiahnutia globálneho riešenie algoritmus založený na metóde vetiev a rezov, pričom pri nekonvexných problémoch bol použitý globálny nelineárny solver BMIBNB. Tento výsledok sa potvrdil aj na probléme získanom v oblasti prediktívneho riadenia.

Bibliography

[1] Judice J.J., Faustino A.M. The Solution of the Linear Bi-Level Programming Problem by Using the Linear Complementarity Problem. *Investigação Operacional*, 8:79–85, 1988.

[2] Adjiman C.S., Androulakis I.P., Floudas C.A. . *AIChE Journal*, 46:1769–1797, 2000.

[3] Gumus Z.H., Floudas CH.A. Global Optimization of Nonlinear Bilevel Programming Problems. *Journal of Global Optimization*, 26:199–219, 2003.

[4] Papamichail I., Adjiman C.S. Global Optimization of Dynamic Systems. *Computers and Chemical Engineering*, 28:403–415, 2004.

[5] Axehill D. *Integer Quadratic Programming for Control and Communication*. PhD thesis, Linköping University, Sweden, 2008.

[6] Anandalingham G., White D.J. A Solution for the Linear Static Stackelberg Problem Uding Penalty Functions. *IEEE Transaction on Automatic Control*, 35:1170–1173, 1990.

[7] Marcotte P., Savard G., Zhu D.L. A Trust Ragion Algorithm for Nonlinear Bilevel Programming. *Operations Research Letters*, 29:171–179, 2001.

[8] Faisca N.P., et al. Parametric Global Optimisation for Bilevel Programming. *Jurnal of Global Optimization*, 38:609–623, 2007.

[9] Floudas C.A. et al. Global Optimization in the 21st Century: Advances and Challenges. *Computers and Chemical Engineering*, 29:1185–1202, 2005.

[10] Hansen P., Jaumard B., Savard G. New Branch-and-Bound Rules for Linear Bilevel Programming. *Society for Industrial and Applied Mathematics*, 13:1194–1217, 1992.

[11] White D.J., Anandalingam G. A Penalty Function Approach for Solving Bi-Level Linear Programs. *Journal of Global Optimization*, 3:397–419, 1993.

[12] McCormick G.P. Computability of Global Solutions to Factorable Nonconvex Programs: Part I - Convex Underestimating Problems . *Mathematical Programming*, 10:147–175, 1976.

[13] L ofberg J., May 2010. http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Solvers.BMIBNB.

[14] Vicente L.N., Savard G., Judice J. Descent Approaches for Quadratic Bilevel Programming. *Journal of Optimization Theory and Applications*, 81:379–399, 1994.

[15] Nguyen T.Q., Bouhtou M., Lutton J-L. D.C. Approach to Bilevel Bilinear Programming Problem: Applicatioin in Telecommunication Pricing. *Optimization and Optimal Control*, 1:211–231, 2003.

[16] Bard J.F. Convex Two-Level Optimization. *Mathemtical Programming*, 40:15–27, 1988.

[17] Bard J.F. Bilevel Linear Programming: Complexity, Equivalence to Minmax, Concave Programs. In *Encyclopedia of Optimization*. 2009.

[18] Edmunds T.A., Bard J.F. Algrithms for Nonlinear Bilevel Mathematical Programs. *IEEE Transactions on Systems,Man, and Cybernetics*, 21:83–89, Janury/February 1991.

[19] Shimizu K., Ishizuka I. , Bard J.F. *Nondifferentiable and Two-Level Mathematical programming*. Kluwer Academic Press, 1997.

[20] Bard J.F., Moore J.T. A Branch and Bound Algorithm for the Bi-Level Programming Problem. *SIAM Jurnal on Scientific and Statistical Computing*, 11:281–292, 1990.

[21] Aiyoshi E., Shimizu K. Hierarchical Decentralized Systems and Its New Solution by a Barrier Method. *IEEE Transaction on Systems, Man, and Cybernetics*, 11:444–449, 1981.

[22] Wolsey L.A. *Integer Programming*. John Wiley & Sons,Inc., 1998.

[23] Bemporad A., Morari M. Control of Systems Integrating Logic, Dynamics, and Constraints. *Automatica*, 35:407–427, 1999.

[24] Jones C.N., Morari M. Approximate Explicit MPC Using Bilevel Optimization.

[25] Lasdon L.S., Warren A.D, Jain A., Ratner M. Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming. *ACM Transactions and Mathematical Software*, 4:34–50, 1978.

[26] Richter S., Jones C.N., Morari M. Real-Time Input-Constrained MPC Using Fast Gradient Methods. In *Conference on Decision and Control, CDC*, Shanghai, China, December 2009.

[27] Scholtes S., Stöhr M. Exact Penalization of Mathematical Programs with Equilibrium Constraints. *SIAM Journal of Control and Optimization*, 37:617–652, 1999.

[28] Hajezi S.R., Memariani A., Jahanshahloo G., Sepehri M.M. Linear Bi-Level Programming Solution by Genetic Algorithm. *Computer and Operations Research*, 29:1913–1925, 2002.

[29] Horst R., Thoai N.V. DC Programming: Overview. *Journal of Optimization Theory and Applications*, 103:1–43, 1999.

[30] Muu L.D., Quy N.V. A Global Optimization Method for Solving Convex Quadratic Bilevel Programming Problems. *Journal of Global Optimization*, 20:1–31, 2001.

[31] Wu S., Chen Y., Marcotte P. A Cutting Plane Method for Linear Bilevel Programs. *System Science and Mathematical Science*, 11:125–133, 1998.

[32] Conn A.R., Gould N.I.M, Toint P.L. *Trust-Region Methods*. SIAM, 2000.

[33] Calvete H.I., Galé C., Mateo P.M. A New Approach for Solving Linear Bi-Level Problems Using Genetic Algrithms. *European Journal of Operational Research*, 188:14–28, 2008.

[34] Floudas Ch.A., Pardalos P.M. *Encyclopedia of Optimization*. Springer, 2nd edition, 2009.

[35] Jeroslow R.G. The Polynomial Hierarchy and a Simple Model for Competitive Analysis. *Mathematical Programming*, 32:146–164, 1985.

[36] Dempe S. A Sipmple Algorithm for thr Linear Bilevel Programming Problem. *Optimization*, 18:373–385, 1987.

[37] Nocedal J. , Wright S.J. *Numerical Optimization*. Springer, 2nd edition, 2006.

[38] Stackelberg H. von. *The Theory of the Market Economy*. Oxford University Press, 1982.

[39] Antoniou A., Lu W. *Practical Optimization, Algorithm and Engineering Applications*. Springer, 2007.

[40] Wikipedia, May 2010. http://en.wikipedia.org/wiki/Branch_and_bound.

[41] Wikipedia, May 2010. http://en.wikipedia.org/wiki/Cutting-plane_method.

[42] Wikipedia, May 2010. http://en.wikipedia.org/wiki/Penalty_method.

[43] Wikipedia, May 2010. http://en.wikipedia.org/wiki/Gradient_descent.

[44] Wikipedia, May 2010. http://en.wikipedia.org/wiki/Trust_region.

[45] Wikipedia, May 2010. http://en.wikipedia.org/wiki/Genetic_algorithm.

[46] Wen U.P., Yang Y.H. Algorithms for Solving the Mixed Integer Two-Level Linear Programming Problem. *Computers and Operation Research*, 17:133–142, 1990.